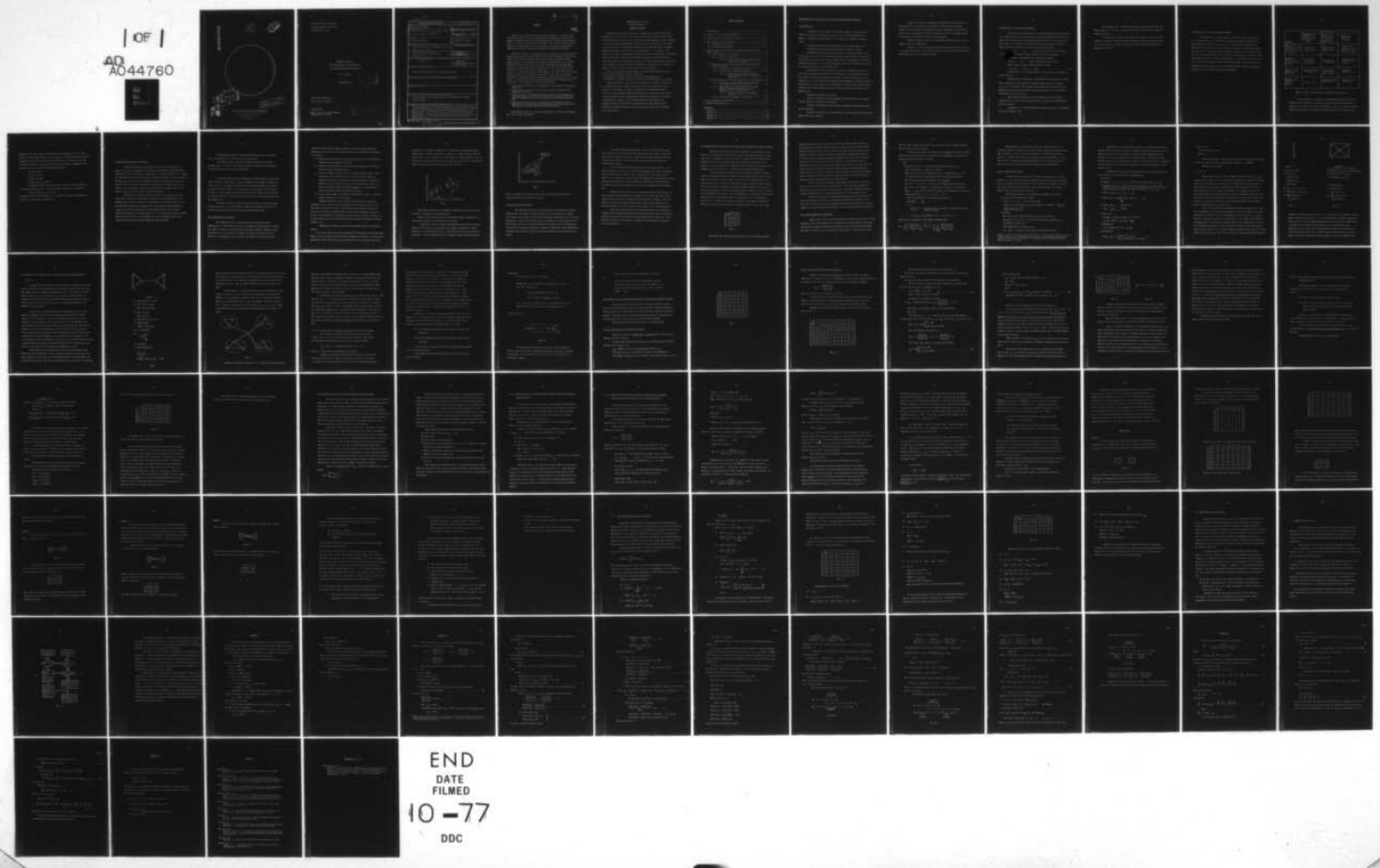
AD-A044 760 ALFRED P SLOAN SCHOOL OF MANAGEMENT CAMBRIDGE MASS C--ETC F/G 12/1
SET DECOMPOSITION: CLUSTER ANALYSIS AND GRAPH DECOMPOSITION TEC--ETC(U)
SEP 77 R C ANDREU N00039-77-C-0255
UNCLASSIFIED CISR-P010-01-01 NL



1 OF 1
AD A044760

END
DATE FILMED
10-77
DDC

12

Center for Information Systems Research
Massachusetts Institute of Technology
Alfred P. Sloan School of Management
50 Memorial Drive
Cambridge, Massachusetts, 02139
617 253-1000

Contract No. N00039-77-C-0255

Internal Report No. P010-01-01

Deliverable No. A002

TECHNICAL REPORT #1

SET DECOMPOSITION: CLUSTER ANALYSIS AND GRAPH DECOMPOSITION TECHNIQUES

R. C. Andreu

September, 1977

Principal Investigator:

Prof. Stuart E. Madnick

Prepared for:

Naval Electronics Systems Command
Washington, D.C. 20360

4B

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: Technical Report #1	2. GOVT ACCESSION NO. / 3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle): Set decomposition: Cluster Analysis and Graph Decomposition Techniques*	5. TYPE OF REPORT & PERIOD COVERED: CISR-P010-01-01, CISR-TR-1
	6. PERFORMING ORG. REPORT NUMBER: P010-01-01
7. AUTHOR(s): Rafael C. Andreu	8. CONTRACT OR GRANT NUMBER(s): N00039-77-C-0255
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Center for Information Systems Research, M.I.T. Sloan School of Management - E53 - 330, Cambridge, MA, 02139	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS: Naval Electronic Systems Command, Washington, D.C. 20360	12. REPORT DATE: September 1977
	13. NUMBER OF PAGES: 80
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) 86p.	15. SECURITY CLASS. (of this report): UNCLASSIFIED
	15a. DECLASSIFICATION DOWNGRADING SCHEDULE: N/A

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Cluster Analysis; Graph Decomposition; Set Decomposition; Software Systems Design.

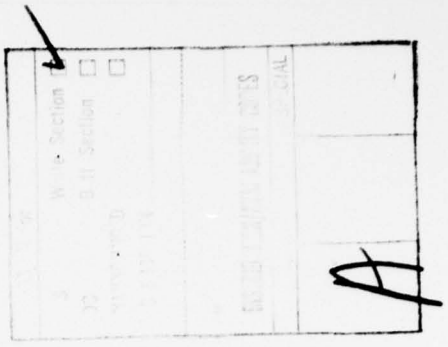
20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The investigation of a systematic approach for the early phases of the system development process generates the problem of decomposing a given set in which interdependencies have been defined among its elements, so as to obtain a collection of subsets as free of interdependencies as possible. This problem is analyzed and solutions proposed; cluster analysis and graph decomposition techniques are applied and shown to possess some similarities which allow to approach set decomposition problems within a unified framework.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

S/N 0102-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)



PREFACE

The Center for Information Systems Research (CISR) is a research center located and managed in M.I.T.'s Sloan School of Management; it consists of a group of Management Information Systems specialists, including faculty members, full-time research staff and part-time students. The Center's general research thrust is to devise better means for designing, generating and maintaining applications software, information systems and decision support systems.

Within the context of the research effort sponsored by the Naval Electronics Systems Command under contract N00039-77-C-0255, CISR proposed to conduct basic research on a systematic approach to the early phases of complex software systems design, one of the main goals being the development of a well defined methodology aimed at explicitly filling the gap between system requirements and program specifications that characterizes most traditional system design strategies. At the heart of such a methodology is the structuring of the initial set of requirements so as to make apparent the design trade-offs existing among its elements; the decomposition of that set into subsets of strongly interdependent requirements which would define a meaningful framework for system design is the main focus of the proposed methodology. The research project is organized so as to investigate the following four areas:

1) Graph-like representation of requirements sets and suitable decomposition techniques,
2) Design and development of a set of software tools to support the set decomposition activity,
3) Identification of a methodology for the assessment of interdependencies among requirements, as well as guidelines for the interpretation of the obtained decompositions and for the coordination of design subproblems, and
4) Experimental application of the methodology and supporting tools to a specific case, with emphasis on recommendadtions for their practical use and comparison with more traditional design approaches.

This document focuses on the activities carried out at CISR to investigate the first area outlined above.

CONTRACT N00039-77-C-0255

Technical Report #1

EXECUTIVE SUMMARY

The main thrust of this research is to explore ways in which to bring more structure to the early stages of the software system design process. These stages are concerned with resolving trade - offs among system requirements so as to identify a collection of design subproblems which can be easily coordinated in the context of the overall design. The approach taken has been to make the trade - offs among requirements explicit and to organize the requirements' set in a graph - like fashion, where nodes correspond to requirements and links to what we call "interdependencies" among them. Strongly interdependent requirements should be considered at the same time for design purposes; subsets of strongly interdependent requirements can be thought of as defining a design subproblem. Consequently, one of the main problems that this research is concerned with is the following:

Given a set of objects (requirements) in which interdependencies have been defined, identify subsets of objects such that:

(a) Elements in the same subset are strongly interdependent, and

(b) Elements in different subsets are as free of interdependencies as possible.

Several approaches to this problem are analyzed and compared in this report. It is found that cluster analysis techniques constitute a good strategy for solving it. Furthermore, a number of interesting correspondencies between cluster analysis and graph decomposition techniques are identified which provide a unified framework where decomposition problems can be analyzed. One of these correspondencies suggests a new clustering algorithm which requires less a priori parameters than other more traditional ones. Experimental results with this algorithm as applied to the problem outlined above will be included in a future report.

TABLE OF CONTENTS

SET DECOMPOSITION: CLUSTER ANALYSIS AND GRAPH DECOMPOSITION TECHNIQUES

1.- Introduction

The purpose of this report is to survey a number of techniques that address the problem of decomposing a given set, in which "interdependencies" among its elements have been defined, into a collection of subsets characterized by:

a) Strong interdependencies among elements of a given subset, and

b) Weak interdependencies among elements of different subsets.

The motivation for this survey is discussed in [Andreu & Madnick 77], where decomposing a set of system requirements into subsets of the characteristics outlined above is presented as a means towards the end of developing a systematic methodology applicable to the system development phase called "preliminary design" or "architectural design" (see [Freeman 76]).

The emphasis of the discussion will be both on techniques proposed to solve the decomposition problem and on the nature of the interdependencies among set elements assumed by the different techniques. Since one of the goals of our research is to investigate ways of assessing interdependencies among the requirements established for a given system, it is appropriate to consider how similar interdependencies have been treated in the past in order to gain insight into how can we approach ours.

The paper is organized as follows:

Section 2 describes the basic framework in which most of the so called "cluster analysis" techniques are organized.

Section 3 discusses how cluster analysis problems are typically defined in that framework.

Section 4 describes two basic approaches to cluster analysis and discusses some of their pros and cons.

Section 5 focuses on a decomposition problem that doesn't quite fit the framework of cluster analysis but which is relevant for our purposes because (i) it represents a starting point for the formulation of our problem, (ii) it can be transformed to fit the framework, and (iii) it provides insight as to how some "clustering algorithms" can be improved.

Section 6 briefly reviews a few techniques appropriate only in very special cases, for completeness.

Section 7, finally, discusses some practical implications of sections 4, 5 and 6, and proposes a strategy for choosing a decomposition technique given the characteristics of the problem at hand.

2.- The basic cluster analysis framework.

The purpose of cluster analysis, as defined by Hartigan ([Hartigan 75]) is to "group similar objects". Whatever the technique employed, this definition implies that information is available regarding the extent to which objects in the initial set are similar or dissimilar. Most cluster analysis techniques assume that such information is available in the following way:

Let $O: \{o_1, \ldots, o_i, \ldots, o_n\}$ be the initial set of (n) objects in which clusters ("groups of similar objects") are to be identified.

Each object is characterized by a set of "attributes",

$X: \{x_1, \ldots, x_j, \ldots, x_m\}$, measured in some scale(s).

Thus, an object $o_i \varepsilon O$ is characterized by a vector

$$X_i: (x_{i1}, \ldots, x_{ij}, \ldots, x_{im}) \qquad \ldots\ldots\ldots\ldots\ldots\ldots (1)$$

In this sense, X_i is a representation of object o_i for the purposes of the analysis.

Cluster analysis algorithms, as discussed in the next section, assume that "similarity" or "dissimilarity" measures between any pair of objects $o_i, o_j \varepsilon O$ are available. These measures are computed from the objects' representation X_i; several ways of doing so have been proposed, as it is pointed out briefly in the next section.

From a broader perspective, in the context of a set of objects O represented by a set of vectors X_i of the form (1), there are two things that can be "clustered":

a) Objects (i.e., identifying groups of similar objects, the objective of cluster analysis), and

b) Attributes (i.e., identifying groups of attributes that define the "main components" of vectors X_i representing objects in some set, the realm of "factor analysis").

In this paper, we focus on techniques that address the first of these problems, although in the final section we suggest that the second can be relevant for our purposes in order to give meaning to the subsets identified in a decomposition process.

3.- Characteristics of dissimilarity measures.

As advanced above, cluster analysis techniques typically assume that an n x n matrix $S(s_{ij})$ of "dissimilarity (sometimes similarity) coefficients", defined between any pair of objects o_i, $o_j \in O$ exists. These coefficients are obtained by manipulating the representations X_i of objects o_i in some meaningful way. The specific strategy employed to compute S depends upon the characteristics of the attributes relevant to the set of objects O. Attributes, in general, can be classified according to two main characteristics: (i) the size of their range sets (i.e., the sets from which attributes draw their values), and (ii) the scale of measurement employed. Table 1, taken from [Anderberg 73], displays one possible classification of attributes and shows some examples.

Scale measurement	Size of range set		
	Continuous: May assume an uncountably infinite number of values	Discrete: May assume a finite (at most countably infinite) number of values	Binary: May assume only two values
Ratio: If $x_A > x_B$, A is x_A/x_B times greater than B and $x_A - x_B$ units greater than B.	Temperature in °K, weight, height, age	Counts such as number of cars, persons, etc.	Unit price of drinks in vending machines, e.g.: cups, 10¢;cans:15¢
Interval: If $x_A > x_B$, A is $x_A - x_B$ units greater than B.	Temperature in °C, specific gravity	Serial numbers, TV channel nos.	How many wives do you have? (only 0 or 1).
Ordinal: Either $x_A > x_B$, $x_A = x_B$ or $x_A < x_B$.	Human judgements of texture, etc.	Military rank, (wide, medium, narrow), etc.	Tall-short, good-bad, etc.
Nominal: Either $x_A = x_B$ or $x_A \neq x_B$	----------	Eye color, place of birth, etc.	Yes-no, on-off, true-false.

Table 1: Cross - classification of attributes with examples.

For the purpose of computing the dissimilarity matrix S, scale conversions may be needed prior to combining the representations of any pair of objects X_i and X_j into an S entry of the form $s_{ij} = f(X_i, X_j)$. Scale conversion techniques applicable to the attributes listed in Table 1 are available (see

[Anderberg 73]). The specific strategy employed to compute S is of no central concern to our discussion here; it will suffice to say that most cluster analysis techniques assume that the dissimilarity coefficients are "metrics", one type of distance functions. Several such functions are discussed in [Anderberg 73]. Metrics are characterized by the following properties:

1) $s_{ij} = 0$ iff $i = j$.

2) $s_{ij} \geq 0 \quad \forall i,j$.

3) $s_{ij} = s_{ji} \quad \forall i,j$.

4) $s_{ij} \leq s_{ik} + s_{kj} \quad \forall i,j,k$.

A distance function having properties 1, 2 and 3 but not property 4 is called a "semimetric". If, on the other hand, property 4 is replaced by

4') $s_{ij} \leq \max(s_{ik}, s_{kj}) \quad \forall i,j,k,$

a distance function with properties 1, 2, 3 and 4' is called an "ultrametric", since 4' is considerably stronger than 4.

4.- Basic cluster analysis approaches.

Once a dissimilarity matrix $S(s_{ij})$ of "distances" between pairs of objects in a set O is available, there are two main approaches, discussed below, that make use of the information in that matrix to identify clusters, i.e., subsets of O characterized by the fact that their elements are similar, or "close together", in some sense. From a slightly different perspective, clusters can be viewed as groups of objects such that the members of a given one are closer to any of the objects in the same cluster than to some other object in another cluster. This view is at the root of many cluster analysis algorithms.

The concept of cluster as decribed above is not very precise, in the sense that no concrete characterization of cluster can be derived from it. For example, how close must an object be to the members of a given cluster for it to be considered an element of the same cluster (as opposed to being assigned to a new cluster? This question hinges on two related aspects of the cluster analysis methodology as it exists today, i.e.: (i) into how many subsets (clusters) should the original set O be decomposed?, and (ii) how coherent should these clusters be? There are two trivial answers to these questions:

(i) The minimum number of clusters that may result is 1, the entire set O; the maximum is n, n clusters with one object each.

(ii) Minimum cluster coherence is achieved with only one cluster (assuming that the set O has more than one element); maximum cluster coherence is achieved with n clusters of one object each.

Of course, neither extreme is meaningful. Cluster analysis is concerned with some middle ground solution. Several parameters can be used to characterize such a solution. For example, we can say that the distance between two elements in any given cluster should not be greater than a given amount, or that we wish to obtain a certain number of clusters. Alternatively, measures can be developed to evaluate, globally, a given decomposition of the set O, and algorithms devised to optimize them.

The cluster analysis techniques currently available can be classified in two main families. They work towards a middle ground solution starting at one of the two extremes just mentioned. We discuss them briefly below.

4.1.- Agglomerative techniques.

The techniques in one of these families are generically called "agglomerative". They start with a set of n one-member clusters and try to reduce the number of clusters as dictated by some meaningful criteria. Typically, these techniques go all the way until the number of clusters is reduced to 1 (the entire set O). The order in which elements are assigned to clusters that will

eventually merge into the complete original set is then used to identify a reasonable set of clusters. The general structure of this family of techniques is as follows:

1.- Begin with n clusters, each containing one object. Let the clusters be labelled with the numbers 1 through n.

2.- Search the dissimilarity matrix for the most similar pair of clusters. Let the choosen clusters be p and q.

3.- Reduce the number of clusters by 1 by merging clusters p and q. Label the product of the merger q (say $q < p$ by convention). Update the dissimilarity matrix to reflect the revised dissimilarity coefficients between the new cluster q and all other existing clusters. (Note, the matrix now contains distances between clusters, not between objects). Delete the row and column pertaining to cluster p.

4.- If the current number of clusters is greater than 1, go to 2. Otherwise stop: all objects are in one cluster.

Several techniques in this family can be developed by varying the procedures used for defining the most similar pair of clusters at step 2 and for updating the matrix at step 3. (Note that distances between clusters are not as well defined as distances between objects are: for example, they can be defined as the maximum, or the minimum, distance between any pair of objects, one in each cluster; or as the distance between the so called "centroids" of the two clusters, etc.)

Agglomerative techniques of this kind are appealing for the following reasons:

- In a sense, they scan the ground between the two extreme solutions mentioned above in such a way that when the process is complete the analyst can decide upon an appropriate middle ground solution (see [Choffray 77] for a method to make such

a decision). For example, consider Fig. 1. The objects represented as points there could be succesively clustered as shown (i.e., first A with B, then this cluster with C, etc.). Cluster "5" is the complete set. But clusters "2" and "4", the only ones existing just prior to the merger that produces "5", form possibly (and intuitively) the best set of clusters for this case.

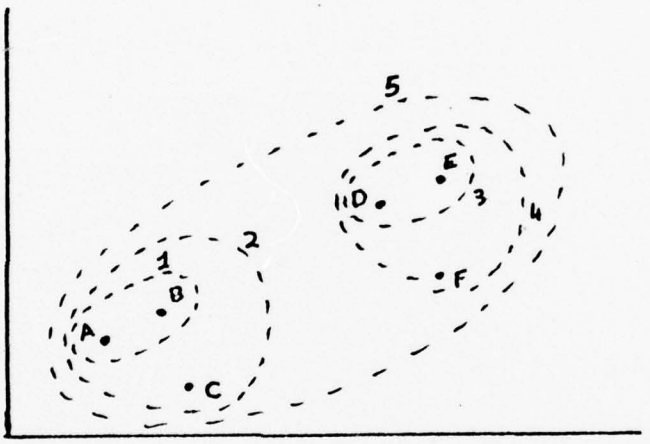


Fig. 1

- They explicitly display a hierarchical arrangement of clusters that can be useful to interpret the chosen partition.

- They scan the ground between the two extremes without enumerating all the possible clusters, so that they tend to be fast.

On the other hand, they make early decisions that are never reconsidered and which may not be appropriate. For example, consider Fig. 2 which represents a two-dimensional case (i.e., two attributes) that can be drawn on the plane. In the circumstances of Fig. 2, objects A and B, the closest pair,

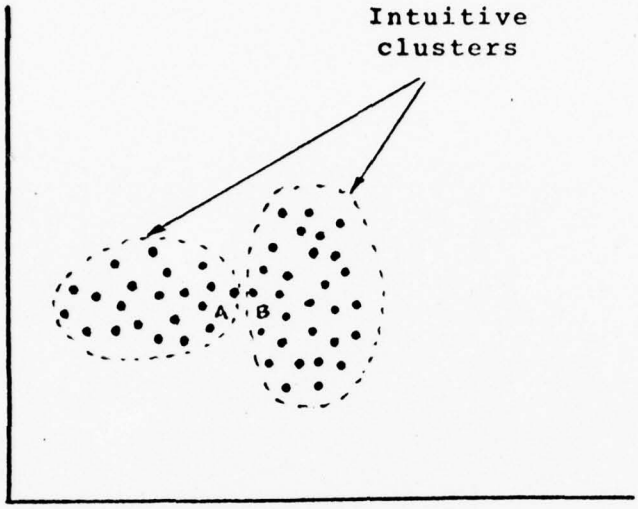


Fig. 2

would be assigned to the same cluster at the very first step, but this is intuitively wrong from looking at the plot.

4.2.- Partitioning techniques.

The other generic family of cluster analysis techniques can be called "partitioning" techniques. The idea behind them is to start with the complete set of objects as a single cluster and then proceed by partitioning it into subsets by applying a series of rules whose goal is to improve the current partition. For this purpose, some kind of "objective function" is used, that is the counterpart of the function employed to decide what clusters to merge next in the agglomerative techniques discussed above. There are two main subfamilies of partitioning techniques:

(a) "True" partitioning techniques, which start off with the entire set of objects as a single cluster and keep partitioning it while an objective function keeps improving. The way in which succesive partitions are generated is not always "general" in the sense that not all possible partitions are considered for adoption.

(b) "Initial partition" techniques, which proceed as follows: (1) An eventual number of clusters k is decided upon a priori, (2) either a k-partition or k so called "leader" objects --each corresponding to one of the eventual clusters-- are identified, and (3) objects are assigned or re-assigned to clusters while improvement results. Several methods for identifying initial partitions or leader objects have been proposed, but no one is recognized as the absolute best.

Techniques in this family are in general more iterative (and thus more time consuming) than agglomerative techniques; on the other hand, they tend to avoid the main problem that characterizes agglomerative techniques: since early decisions regarding which object goes to what cluster may be revised as the algorithm proceeds, they have a chance to be changed if this seems appropriate.

The central idea driving most of these techniques can be expressed in a mathematical programming framework; however, solving the resulting optimization problem is often impractical because of its characteristics (large size, binary decision variables, non-linearity). An example of this situation is presented in the next section.

5.- Graph decomposition problems and techniques; similarities to cluster analysis.

While the decomposition techniques outlined in the preceding sections work with a "dissimilarity matrix" whose entries meet certain conditions (basically they are assumed to be metrics), there is a general form of decomposition problem that does not meet these conditions but which is of interest to us: Consider a situation where pairs of objects in the initial set O are known to be related in one of two ways, e.g., they are either related or unrelated.

If we try to put this problem in the framework described in sections 2 and 3, we immediately think of a distance matrix whose entries are of only two types: given a pair of objects, they are either "close" (related) or not, so that matrix entries can take only two possible values. Assume we assign a value of 1 to the entries corresponding to pairs of related objects and a value of 0 to those corresponding to pairs of unrelated objects. A matrix of this form would be a "similarity" matrix rather than a dissimilarity one. Switching the assignment strategy for 1's and 0's would result in a matrix more like a dissimilarity matrix, but its entries would not meet, in general, the conditions needed for them to be metrics (see, for instance, the matrix in Fig. 3: $s_{12} > s_{13} + s_{32}$).

i \ j	1	2	3
1	0	1	0
2	1	0	0
3	0	0	0

Fig. 3

This means that such a matrix cannot be used in the cluster analysis

techniques discussed above. Nevertheless, we have intuitive feelings about how the decomposition problem should be attacked in this case: The eventual clusters should be groups of objects whose elements are strongly pairwise related, while elements in different subsets should have few pairwise relationships. This can be formalized and a strategy devised for decomposing the initial set of objects into subsets with these characteristics. Moreover, it is possible to derive dissimilarity matrices from the initial binary matrix in such a way that the resulting entries are metrics. This means that we can look at this problem from either the point of view of traditional cluster analysis or from that of methodologies appropriate for binary matrices; as it turns out, this dual possibility suggests ways of improving techniques in both settings.

In this section we discuss the binary matrix case with the convention that 1's are assigned to pairs of related objects. This convention permits the representation of the initial set of objects as a graph whose "adjacency matrix" (see [Deo 74])corresponds to the similarity matrix in the terminology above.

Two ways of solving this decomposition problem will be briefly described; two strategies for deriving a similarity matrix meeting the metric conditions will be proposed, and the decomposition techniques applicable to the latter compared with those suitable for the former --both can benefit from the comparison.

5.1.- Graph decomposition techniques.

Given a set of objects whose interdependencies are characterized by a binary similarity matrix, the decomposition problem becomes a graph decomposition problem. As described in [Andreu & Madnick 77], a measure M can be developed that evaluates how "good" a partition of the original set into subsets is. This measure is such

that its value is higher the better the partition, so that it should be maximized across all possible partitions.

If the eventual number of subsets is set a priori (as in many non-agglomerative clustering techniques), the problem can be formalized as a non-linear integer programming problem, as follows.

Let:

- $S(s_{ij})$ be the (n x n) similarity (graph adjacency) matrix, i.e.:

  s_{ij} = 1 iff objects o_i and o_j are related,

  (for the purposes of the formulation below, we assume that s_{ij} = 0 if i=j, although this is counterintuitive and will be changed later)

- K be the eventual number of clusters (set a priori),

- g_{ki} (k=1, ..., K; i=1, ..., n) be a set of binary decision variables;

  g_{ki} is set to 1 if object o_i is assigned to cluster k, to 0 otherwise.

- $M(g_{11}, \ldots, g_{Kn})$ be the value of the measure M corresponding to the partition defined by g_{ki} (k=1, ..., K; i=1, ..., n).

Then the decomposition problem can be written as:

$$\text{Max } M(g_{11}, \ldots, g_{Kn})$$

s.t.:

$$\sum_k \sum_i g_{ki} = n \quad \text{(to ensure that each object is assigned to one and only one cluster)}$$

$$1 \geq g_{ki} \geq 0, \; g_{ki} \text{ integers, } k=1, \ldots, K; \; i=1, \ldots, n,$$

where M can be expressed as (see [Andreu & Madnick 77]):

$$M = 2 \sum_{k=1}^{K} \frac{\sum_j \sum_i g_{ki} g_{kj} s_{ij} - g_{ki} + 1}{(\sum g_{ki})^2 - \sum g_{ki}} - \sum_{k=1}^{K} \sum_{\substack{k'=1 \\ k' \neq k}}^{K} \frac{\sum_j \sum_i g_{ki} g_{k'j} s_{ij}}{(\sum g_{ki})(\sum g_{k'i})}$$

This formulation is impractical given the characteristics of the resulting problem: non-linearity and a large number of binary decision variables (nK). Although it conveys the general structure of the graph decomposition problem, it suggests that heuristic approaches can be more appropriate. In the next section we explore one such approach which is also useful to identify a way for putting the graph problem in the framework of cluster analysis.

5.1.1.- A heuristic approach.

In this subsection we describe a heuristic approach that has proven effective in a few graph decomposition problems where we have actually employed it. It is "subgraph strength" driven, in the sense that it attempts to identify the "core" of subsets of nodes likely to have high "strength" (see [Andreu & Madnick 77] for a definition of this term).

For exposition purposes, the following terminolgy will be used:

Let the pair (O,A) represent a graph:

- O: $\{o_i,\ i=1, \ldots, n\}$, the set of (n) nodes,
- A: $\{a_{ij},\ a_{ij}=1$ if nodes o_i and o_j are related, $=0$ otherwise$\}$; here we assume $a_{ij}=1$ when $i=j$ (*).

We define:

- The "core set" CS_i associated with a node o_i to be the set $CS_i: \{o_j \mid o_j$ s.t. $a_{ij}=1\}$, i.e., the set of all nodes related to o_i, including o_i itself, and
- The "connectivity" of node o_i to be $c_i = |CS_i| - 1$, where by $|X|$ we mean the dimension of set X.

(*) If we think of A as the graph adjacency matrix, this would mean that all the nodes have "self-loops" ([Deo 74]); in our context this is intuitively correct: a node is related to itself.

Intuitively, we are interested in nodes $o_i \in O$ with high connectivity and whose associated cores sets do not interfere too much with one another, since those are likely to be at the kernel of subsets of nodes whose elements are strongly related. Once a number of such "kernel subsets" are identified, the remaining nodes can be assigned to the subsets in which they best "fit", according to some criteria consistent with the maximization of the overall measure M (see section 5.1.1).

Following this intuition, the identification of kernel subsets can be done iteratively using the procedure specified below:

0) Set J=0.

1) Compute $c_i \ \forall o_i \in O$. If $c_i = c_j \ \forall i,j$, set J=J+1; KESU(J) = O; stop.

2) Consider the $\underline{k}$ (> 1, a number specified $\underline{a\ priori}$; see the end of this section for considerations about its value) nodes with highest c_i. Without loss of generality, assume that these are the nodes $o_1, \ldots, o_k$.

3) Determine CS_i for $o_i \in \{o_1, \ldots, o_k\}$.

4) Compute $KS_i = (CS_i \cap [\bigcup_{\substack{j=1 \\ i \neq j}}^{k} CS_j]) \ \forall o_i \in \{o_1, \ldots, o_k\}$.

5) Select $o_p \in \{o_1, \ldots, o_k\}$ such that

$$KS_p = \min_{i=1, \ldots, k} (|KS_i|)$$

6) Set J=J+1;

If $|KS_p| = |CS_p|$, set KESU(J) = O and stop,

else set $KESU(J) = o_p \cup [CS_p - KS_p]$.

7) Recompute O:

$O = O - KESU(J)$; if $|O| = 0$, stop.

8) Recompute A:

$$A: \{a_{ij} \mid a_{ij} = \begin{cases} \text{old } a_{ij} & \text{iff } o_i, o_j \in O \\ \text{mark it "nonexistent"} & \text{otherwise} \end{cases} \}$$

9) k = k - 1;

If k > |O|, set k = |O|;

Go to 1.

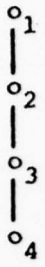
Once this procedure is executed (stopping points are possible at steps 1, 6 and 7), we are left with J kernel subsets KESU(1), ..., KESU(J).

Two cases are possible:

1) J = 1.

This basically tells us that no subset of nodes stands out as a clear candidate kernel subset, i.e., the graph has no meaningful structure. For example, Figures 4 and 5 depict two cases where this situation arises (the execution of the procedure described above is summarized next to each graph). The result is intuitively correct in both cases, particularly for the graph in Fig. 5. The value assumed for the a priori parameter k, however, determines the result which is achieved. We will have something to say about such a value at the end of this section; an intuitive interpretation of the behavior generated by different k values is presented later in section 5.2.2.

Fig. 5 also illustrates the rationale behind the stopping rule at step 1 of the procedure, which is not as intuitive as the ones at steps 6 and 7. The rationale is as follows: If, for a given graph it turns out that $c_i = c_j \ \forall i,j$, there is no apparent structure in it. Note that $c_i = c_j =$ (say) $a \ \forall i,j$ implies that every node in the graph has a links incident to it (incidentally, note also that this means that there are (na/2) links in the graph, so that this circumstance can only occur if $na = \dot{2}$ --even--, which is indeed the case in Fig. 5): thus, there is no way that such links can be arranged so as to display clearly separable



Assume k = 4.

1.- $c_1=c_4=1$; $c_2=c_3=2$.

3.- $CS_1=\{o_1,o_2\}$

$CS_2=\{o_1,o_2,o_3\}$

$CS_3=\{o_2,o_3,o_4\}$

$CS_4=\{o_3,o_4\}$

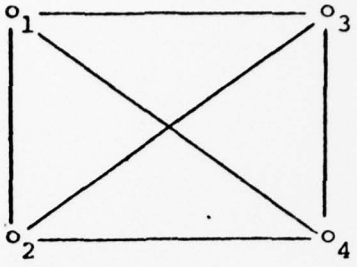
4.- $KS_i = CS_i$, i=1,2,3,4.

5.- (Arbitrarily) $KS_p=KS_1=CS_1$

6.- $|KS_p|=|CS_p|$: Stop;

J=1, KESU(1)= 0

Fig. 4



Assume k = 2.

1.- $c_1=c_2=c_3=c_4=3$; Stop: KESU(1)=0.

If we didn't stop here, we would proceed:

2.- (Arbitrarily) pick o_1,o_2.

3.- $CS_1=CS_2=\{o_1,o_2,o_3,o_4\}$

4.- $KS_1=KS_2=CS_1=CS_2$

5.- (Arbitrarily)

$KS_p = KS_1 = CS_1$

6.- $|KS_p| = |CS_p|$: Stop;

J=1, KESU(1)= 0

Fig. 5

subgraphs. An extreme case occurs for a = n-1 (as in Fig. 5); it implies that the graph is completely connected, i.e., we can interchange the labels of any pair of nodes and still obtain the same adjacency matrix, which obviously suggests that no single node is different in any respect, as far as graph structure is concerned, from any other. The only reasonable alternative in this case is to consider the entire graph to be a single subgraph. The stopping rule at step 1 makes this decision early, thus saving computations (note, for instance, that

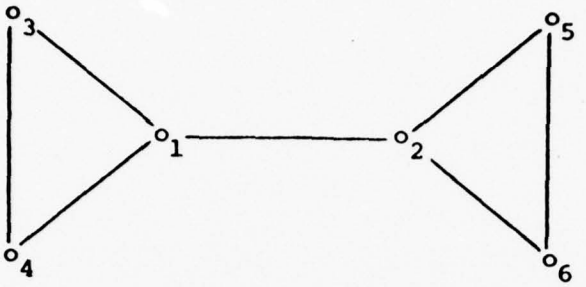
if we do not stop at step 1 in Fig. 5 we still reach the same conclusion).

2) J > 1.

In this case, some subsets of nodes stand out as "good" kernel subsets that can subsequently be completed with the remaining nodes if they do not span the complete set O. An example is presented in Fig. 6. It results in two kernel subsets, namely, $\{o_1, o_3, o_4\}$ and $\{o_2, o_5, o_6\}$, that in this case span the complete set O and thus represent a partition already. This partition turns out to be the best of all possible partitions of that graph as evaluated by the measure M introduced before.

Now we turn to a discussion about $\underline{k}$, the parameter that has to be specified a priori in order to use the procedure described above. Consider again Fig. 4: we assumed $\underline{k} = 4$; this led to the identification of a single subgraph, the entire original graph. Had we taken $\underline{k} = 2$, the procedure would have produced two kernel subsets of nodes: $\{o_1, o_2\}$ and $\{o_3, o_4\}$, that already represent a partition. As measured by M, this is the best two way partition of that graph, but still inferior to the "no partition" solution. However, from a strict graph structure standpoint, the two way partition is superior, since it is apparent that the subgraphs $\{o_1, o_2\}$ and $\{o_3, o_4\}$ are structurally identical and thus display some special configuration inherent to the initial graph (see section 5.2.2 below for a more intuitive interpretation of this).

What can then be said about possible values for $\underline{k}$?. An obvious upper bound is $\underline{n}$, the total number of nodes in the initial graph. In most cases, however, choosing k=n will result in a single kernel subset: the entire graph. The reason for this is that if we take into consideration the "core sets" of all nodes,



Assume k = 2.

1.- $c_3=c_4=c_5=c_6=2$; $c_1=c_2=3$

3.- $CS_1 = \{o_1, o_2, o_3, o_4\}$

$CS_2 = \{o_1, o_2, o_5, o_6\}$

4.- $KS_1 = \{o_1, o_2\}$

$KS_2 = \{o_1, o_2\}$

5.- (Arbitrarily) $KS_p = KS_1$

6.- $|KS_p| < |CS_p|$;

KESU(1) $= \{o_1, o_3, o_4\}$

7.- $O = \{o_2, o_5, o_6\}$

8.-

1.- $c_2=c_5=c_6=2$

3.- $CS_2 = \{o_2, o_5, o_6\}$

$KS_p = CS_2$

↓

KESU(2) $= \{o_1, o_3, o_4\}$; stop

Fig. 6

chances are that their interferences will be so considerable that no one of them will stand out as clearly isolated; the procedure will tend to stop at step 6 (i.e, the best core set is completely interferred by all the others). This is illustrated in Fig. 5. Thus, k should not be set as high as n or any close value.

A lower bound is 1, but in general this is not an appropriate value either. The reason is that it can lead to very undesirable results because basically it is equivalent to picking a good core set without worrying about whether it interferes with others or not. For example, consider the graph in Fig. 7. Taking k=1 results in the kernel subset $\{o_1, o_2, o_5, o_8, o_{11}\}$, likely to preclude from further consideration the intuitive partition shown in the figure.

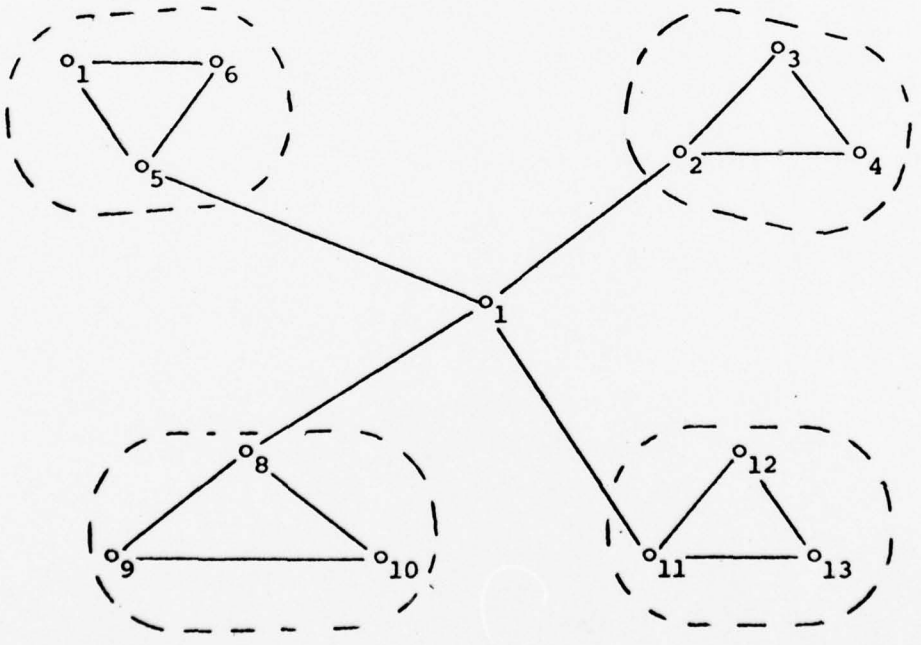


Fig. 7

Therefore, k should be such that $1 < k < n$. This is not a very strong

assertion. Our experience indicates that k can be set to a value somewhat higher than the expected number of subgraphs. Furthermore, varying k has the following intuitive effect: the greater k, the more "conservative" we are, in the sense that less subgraphs will be identified because we consider interferences among many core sets. Experimenting with a few k values and looking for stability in the results is usually a good strategy.

A less ad-hoc strategy for choosing k is as follows: After step 1, order the c_i's (i=1, ..., n) in decreasing order (this has to be done anyway). Then, pick the node with maximum c_i and all the following ones that have a c_i higher than a given percentage (< 100) of that maximum. This percentage figure still has to be specified a priori, but it seems a more unified measure of "how risky" we wish to be. For example, picking 80% leads to k=2 for the examples in Figs. 4 and 6. Any percentage figure leads to k=4 for Fig. 5, which is consistent with the fact that the graph has no apparent subgraphs in the sense we are interested in here.

5.1.1.1 - Comments on a possible generalization of the core set concept.

At this point, a possibility for generalizing the core set concept comes to mind. It is worth investigating, briefly, what happens if we genralize the definition of core set CS_i to read as follows:

$$CS_i: \{o_i, o_j, \text{ s.t. minimum path } (o_i \rightarrow o_j) \leq p;\ p \geq 1\} \quad ,$$

where p = 1 would produce the definition given before.

Although this definition seems intuitively sound, a more careful investigation shows that it makes no good sense in general. To see why, it is useful to view the concept of core set from the following perspective:

The construction of a core set in a given graph is equivalent to marking each node in the graph in a binary fashion (i.e., a specific node either belongs to the core set of it doesn't). In this sense, two nodes are considered equivalent (with respect to the node whose core set is being constructed, which can be called the "core set node", CSN) when the minimum paths from the CSN to either of these two nodes is $\leq p$. From a strict graph structure viewpoint, this is wrong, unless $p = 1$. The reason is that if $p > 1$ the two paths may in fact (i) differ in length or (ii) be completely different even if their lengths are equal. Thus, taking $p > 1$ in the definition of core set loses information because it leads to summarize in a binary output a comparison that can produce more than two outcomes in general.

Therefore, if we wish to generalize the definition of core set to apply for $p > 1$ we should take into account all possible comparison outcomes and make two nodes equivalent with respect to a third (the CSN), in the sense above, only when these two coincide exactly, i.e., when:

(a) The minimum path length from either node to the CSN is the same; and

(b) The two minimum paths are exactly the same, except for the last node.

It can be shown in general that these two conditions imply taking $p = 1$, as illustrated below.

Since condition (b) implies condition (a), all we need to prove is the following:

Proposition

If in a given graph it is true that:

Minimum path ($o_i \rightarrow o_j$) $\equiv$ Minimum path ($o_i \rightarrow o_k$) (except for the last node), then either

1) o_j, o_k are both adjacent to o_i, or

2) o_i is adjacent to neither o_j nor o_k.

(i.e., o_j and o_k are made equivalent by assigning them to CS_i only when they both belong to the CS_i defined with p = 1).

Consider Figure 8:

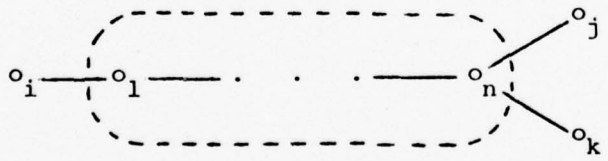


Figure 8

For the proposition condition to be true, the path inside the dotted circle must be part of both the minimum path ($o_i \rightarrow o_j$) and the minimum path ($o_i \rightarrow o_k$). If we assume this to be true, only two cases need to be considered. Either:

- The circled path is empty, which implies (1) above;

or

- The circled path is non-empty, which implies (2) above because if either o_j or o_k (or both) were adjacent to o_i, the circled path would not be part of a path of minimum length. Q.E.D.

5.2.- Putting the graph decomposition problem in the cluster analysis framework.

As was mentioned earlier in section 5, graph decomposition problems are not readily approachable by means of cluster analytic techniques, because the adjacency matrix of a graph fails to meet the conditions assumed for similarity matrices. Since many cluster analytic techniques already exist, resolving this incompatibility would permit the use of this available knowledge.

There are at least two ways of doing so, as described below.

5.2.1.- "Minimum path" dissimilarity measures.

One way is to derive a dissimilarity (distance) matrix from the graph adjacency matrix as follows:

Using the same notation as in section 5.1.2 and calling the resulting distance matrix $S(s_{ij})$, let

s_{ij} = Number of links in the minimum path from o_i to o_j.

The resulting s_{ij}'s can be shown to be metrics (see Appendix I).

For example, the graph in Fig. 6 results in the S matrix shown in Fig. 9.

i \ j	1	2	3	4	5	6
1	0	1	1	1	2	2
2	1	0	2	2	1	1
3	1	2	0	1	3	3
4	1	2	1	0	3	3
5	2	1	3	3	0	1
6	2	1	3	3	1	0

Fig. 9

5.2.2.- "Connectivity" dissimilarity measures.

Another way of deriving a distance matrix for the nodes in a graph is inspired by the concept of "core set" introduced in the heuristic approach discussed in section 5.1.2. Entries in the distance matrix can be computed as follows:

$$s_{ij} = 1 - \frac{|CS_i \cap CS_j|}{|CS_i \cup CS_j|} \quad ,$$

where CS_i is the core set of node o_i (see 5.1.2).

It can be shown (see Appendix II) that the s_{ij}'s so defined meet conditions 2, 3 and 4 of section 3, which characterize metric distances. Condition 1 is not met in general but the strategy is still useful as we discuss below.

Consider, for example, the graph in Fig. 6. Its associated S matrix is shown in Fig. 10.

i \ j	1	2	3	4	5	6
1	0	0.66	0.25	0.25	0.83	0.83
2	0.66	0	0.83	0.83	0.25	0.25
3	0.25	0.83	0	0	1	1
4	0.25	0.83	0	0	1	1
5	0.83	0.25	1	1	0	0
6	0.83	0.25	1	1	0	0

Fig. 10

The following circumstances can be noticed in Fig. 10:

i) The rows (or columns) corresponding to o_3 and o_4 and to o_5 and o_6 are exactly identical.

ii) $s_{34}=s_{56}=0$, which violates metric property 1 (see section 3).

It can be shown in general that these circumstances, if they arise, do so for the same pairs of related nodes, i.e.:

$$\left.\begin{array}{l} s_{ik} = 0 \Leftrightarrow s_{ij} = s_{kj} \quad \forall j \\ s_{ik} = 0 \Rightarrow o_i \text{ and } o_k \text{ are related} \end{array}\right\} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (a)$$

To show this, we proceed as follows:

1) $s_{ik} = 0 \Rightarrow$ (by definition of s_{ik}) $\dfrac{|CS_i \cap CS_k|}{|CS_i \cup CS_k|} = 1$, i.e.:

$|CS_i \cap CS_k| = |CS_i \cup CS_k|$, which is only possible when $CS_i \equiv CS_k$.

This implies $a_{ik} = 1$, i.e., nodes o_i and o_k are related: otherwise, $o_k \notin CS_i$ while --by definition of CS_k-- $o_k \in CS_k$, so that $CS_i \equiv CS_k$ would not hold.

Thus, $s_{ik} = 0 \Rightarrow \begin{cases} CS_i \equiv CS_k \\ o_i \text{ and } o_k \text{ are related.} \end{cases}$

Now, by definition of s_{ij} and s_{ik},

$$s_{ij} = 1 - \frac{|CS_i \cap CS_j|}{|CS_i \cup CS_j|} \; ; \; s_{kj} = 1 - \frac{|CS_k \cap CS_j|}{|CS_k \cup CS_j|} \quad \forall j.$$

But if $CS_i \equiv CS_k$, then $s_{ij} = s_{kj} \; \forall j$, and therefore

$$s_{ik} = 0 \Rightarrow \begin{cases} s_{ij} = s_{kj} \quad \forall j \\ o_i, o_k \text{ are related} \end{cases} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (b)$$

2) If we assume that

$s_{ij} = s_{kj}$ $\forall j$, this certainly holds for $j = k$:

$s_{ik} = s_{kk}$.

But $s_{kk} = 0$, and so $s_{ik} = 0$.

Thus,

$s_{ij} = s_{kj}$ $\forall j \Rightarrow s_{ik} = 0$ ($=s_{ki}$ because S is symmetric) . . . . . .(c)

Equations (b) and (c) together produce equation (a). Q.E.D.

These are very nice properties of the resulting matrix S:

If it turns out that for some pair of nodes o_i and o_k, $s_{ik} = 0$ (i.e., $CS_i \equiv CS_k$), then it is true that $s_{ij} = s_{kj}$ $\forall j$. In other words, nodes o_i and o_k are equivalent with respect to the rest of the graph as described by the matrix S. As far as we consider this matrix an appropriate distance matrix for cluster analysis purposes, the two nodes in that pair can be collapsed into one. This is intuitively appealing since (i) $s_{ik} = 0$ (as $s_{ii} = 0$ $\forall i$) , and (ii) o_i and o_k are related; in fact, it suggests that these nodes should be put in the same subgraph in an eventual partition.

Thus, whenever we come across a $s_{ik} = 0$, we can delete one of the nodes from further consideration, and assign it to whatever subgraph the other one ends up in.

In the case of the graph shown in Fig. 6, doing so with the pairs (o_3, o_4) and (o_5, o_6), and deleting o_4 and o_6, produces the distance matrix S shown in Fig. 11a, corresponding to the "collapsed" graph depicted in Fig. 11b.

i \ j	1	2	3(4)	5(6)
1	0	0.66	0.25	0.83
2	0.66	0	0.83	0.25
3(4)	0.25	0.83	0	1
5(6)	0.83	0.25	1	0

Fig. 11a

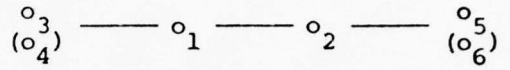


Fig. 11 b

If this is done for all the possible pairs of nodes whose computed distance is zero, the S matrix is reduced in size, its entries are made to meet the first condition for them to be metrics (see section 3), and preliminary clustering is performed.

With the matrix of Fig. 11a, most clustering algorithms would cluster $o_3(o_4)$ with o_1 and $o_5(o_6)$ with o_2, the best partition for the graph of Fig. 6.

From a more general standpoint, it is interesting to observe that, in a sense, the procedure described above treats the nodes of a graph as described by their corresponding rows (columns) of the associated graph adjacency matrix; in other words, it considers each node as described by a set of n attributes: its relationships with all the nodes in the graph. Consequently, the resulting distances (entries in the distance matrix S) tell us something about the overall structure of the graph that was not clearly apparent in the original data (e.g., "collapsable" nodes). This is in contrast with traditional distance matrices, which

do not display such a structure as directly. In this sense, the results obtained when setting k=2 in the procedure of section 5.1.2 as applied to the graph in Fig. 4 (i.e., the two way partition $\{o_1, o_2\}$, $\{o_3, o_4\}$) reflect more accurately the inherent structure of that graph. As it turns out, the strategy proposed above resembles one described in [Curry 76] for computing distances between two objects characterized by a set of attributes measured in a binary scale. The difference with respect to our approach is that we consider the attributes to be the relationships (i.e., similarities) between objects (nodes) that characterize the actual set (graph) under decomposition, as opposed to some external set of attributes. This suggests that perhaps a similar strategy could be employed in general, to make the intrinsic structure of the initial set more apparent in the matrices used in cluster analysis, be they binary or not; we shall explore this possibility in sections 5.3.2 and 5.3.3 below.

Finally, note that the resulting distance matrix is normalized: all its entries assume values in the interval [0,1].

5.2.2.1. An alternative approach to compute distance matrices when preclustering takes place.

The strategy proposed in the preceding section for computing dissimilarity (distance) matrices is basically a two-step process as follows:

(1) Compute distances between all pairs of nodes.

(2) Check whether any computed distance is zero. If so, delete the row and column corresponding to the zero entry in the distance matrix.

Note that since distances are computed before collapsing nodes, they correspond to the structure of the original graph: collapsed nodes are taken as separate nodes for the purposes of distance computation.

It can be shown in general, however, that any two nodes o_i and o_k such that $s_{ik} = 0$ either appear both or do not appear at all in a given core set CS_p:

By the definition of core set, it follows that

$$o_i \in CS_k \Leftrightarrow o_k \in CS_i ,$$

because $o_i \in CS_k \Rightarrow a_{ik} = 1 = a_{ki} \Rightarrow o_k \in CS_i$ and vice versa.

So, since $s_{ik} = 0 \Rightarrow CS_i \equiv CS_k$, it follows that:

If $s_{ik} = 0$,

a) $\underline{o_i \in CS_p} \Rightarrow o_p \in CS_i \Rightarrow o_p \in CS_k \Rightarrow \underline{o_k \in CS_p}$, and

b) $\underline{o_i \notin CS_p} \Rightarrow o_p \notin CS_i \Rightarrow o_p \notin CS_k \Rightarrow \underline{o_k \notin CS_p}$, q.e.d.

This suggests that entries in the distance matrix can be alternatively computed by considering collapsed nodes as single nodes, i.e., by basically reversing the order ot steps (1) and (2) above. The resulting entries in the distance matrix will still be metrics, and they won't coincide, in general, with entries computed as described above.

A simple example will make this point clearer: Consider the graph in Figure 6 (page 22). The distance matrix computed as proposed in the preceding section (after collapsing nodes) is that in Figure 11a.

The alternative distance matrix computation would proceed as follows:

Upon recognition that nodes o_3 and o_4, and o_5 and o_6 collapse, the graph in Fig. 2 is in fact reduced to that shown in Fig. 11b.

The core sets for this graph are as follows:

CS_1: $\{o_1, o_2, o_3(o_4)\}$

CS_2: $\{o_1, o_2, o_5(o_6)\}$

$CS_{3(4)}$: $\{o_1, o_3(o_4)\}$

$CS_{5(6)}$: $\{o_2, o_5(o_6)\}$

so that the resulting distance matrix would be that in Fig. 12.

	1	2	3(4)	5(6)
1	0.00	0.50	0.33	0.75
2	0.50	0.00	0.75	0.33
3(4)	0.33	0.75	0.00	1.00
5(6)	0.75	0.33	1.00	0.00

Figure 12

The question thus arises: Which of the two strategies should be employed to compute distance matrices for a graph?

First of all, note that the two strategies trivially coincide when no collapsing of nodes occurs. On the other hand, it seems intuitively appealing to use the first strategy when collapsing does occur, since the distances so computed reflect more accurately the structure of the original graph. For example, compare the graphs in Figs. 11b and 4. Using the second strategy would produce the same distance matrix for these graphs. This is intuitively wrong, since we know that in Fig. 11b, node o_3 (o_4) corresponds to two original nodes. The fact that $d_{1,(3,4)}$ in Fig. 11a is less than $d_{1,(3,4)}$ in Fig. 12 confirms this intuition: the two collapsed nodes o_3 and o_4, both close to o_1 in the original graph (Fig. 6) in fact "pull" o_1 closer to the node resulting from collapsing o_3 and o_4 than it would otherwise be, in a graph such as that of Fig. 4.

For these reasons, we will predominantly use the first strategy to compute distance matrices for graphs in which collapsing occurs.

5.3.- Graph decomposition techniques applied to cluster analysis.

Section 5.2 discussed some strategies for constructing a metric distance matrix from a graph adjacency matrix, so that the graph decomposition problem can be treated in a cluster analytic framework. In this section we explore the opposite move, i.e., is it possible to represent a cluster analysis problem as a graph decomposition one? The motivation for this is that if such a transformation were possible, some of the techniques suitable for graph decomposition problems like the heuristics described in section 5.1.2 could be employed in cluster analytic algorithms and their effectiveness investigated.

In general, and from a strict point of view, the answer to the above question is no. Of course, some distance matrices can be made to correspond to a graph adjacency matrix (e.g., we know that the matrix in Fig. 8 corresponds to the graph in Fig. 6), but there is no straight forward algorithm that we know of which would permit us even to assess whether or not a given distance matrix corresponds to some graph, in general, let alone what that graph would be. Nevertheless, we still feel that some insight can be gained from exploring this move. The next section proposes a way of applying the ideas behind the heuristics of section 5.1.2 to cluster analysis problems. Prior to that discussion, however, we point out a crude method for putting cluster analysis problems in graph form. This approach, which was proposed in [Hubert 74], works as follows:

Given an nxn distance matrix $S(s_{ij})$, choose a threshold value $T > 0$ and define

$$s'_{ij} = \begin{cases} 0 \text{ if } s_{ij} \geq T \\ 1 \text{ if } s_{ij} < T \end{cases}$$

We can then think of the matrix $S'(s'_{ij})$ as the adjacency matrix of a graph in which pairs of related nodes correspond to objects in the original problem whose distance is less than T. Of course, this transformation losses a great deal of information, because all the resulting s'_{ij}'s are either 0 or 1, while the original s_{ij}'s, in general, would vary more smoothly in a wider interval; furthermore, different T values can produce different graphs. However, this strategy can be appropriate when the original entries in S take values that are either very high or very low, by choosing a T value in between. The resulting graph can be drawn and maybe even visual insight employed to identify an initial partition.

A more sophisticated approach could proceed as follows:

1) Choose several T values T_1, T_2, ..., T_m.

For each T value,

2) Derive a graph as described above.
3) Using one of the transformations in 5.2.1 or 5.2.2, construct a distance matrix for the graphs generated in 2.
4) Conduct a statistical comparison test between the distance matrices computed in 3 and the original one.
5) Retain the T value that produced the graph which resulted in the best fit in 4.

This, however, would probably be very time consuming and not worth the effort if the output was to be only an initial partition. A more direct approach that takes advantage of the heuristics discussed in section 5.1.2 is proposed in the next section.

5.3.1 - A strategy for constructing initial partitions in non-agglomerative cluster analysis.

As was pointed out in section 4.2, no particularly "good" methods to identify initial partitions for non-agglomerative cluster analysis techniques are available. We think that a potentially appropriate method can result from the application of a strategy similar to that employed to construct kernel subsets in the graph case (see 5.1.2).

We describe one such strategy below. Its parallelism with the procedure in 5.1.2 should be apparent.

Given an n×n distance matrix $S(s_{ij})$ defined over a set of n objects $O: \{o_1, o_2, \ldots, o_n\}$, this procedure would proceed as follows:

1) Pick a threshold value T, $0 < T < \max (s_{ij} \in S)$.

2) Set $J = 0$.

3) For each $o_i \in O$, compute:

$CS_i = \{o_j | o_j \in O \text{ and } s_{ij} \leq T\}$.

$c_i = |CS_i| - 1$.

4) Consider the $k > 1$ objects with highest c_i. Without loss of generality, assume that these objects are $o_1, \ldots, o_k$.

From this point on, this procedure is very similar to one proposed in [Astrahan 70] for identifying leader objects (see 4.2). Since Astrahan's technique is considered to be one of the most intuitively sound for these purposes (see [Anderberg 73]), we believe that the one described above would also perform well; furthermore, it requires fewer a priori parameters than Astrahan's (only two, T and k, as compared with three requires by Astrahan's method).

5.3.2 - Working with the similarity matrix as a whole prior to applying clustering algorithms; normalization of distance matrices.

In this section we propose a strategy for transforming a given similarity (or dissimilarity) matrix into a new, normalized one, by means of a procedure similar to that described in section 5.2.2 for transforming adjacency matrices into distance matrices.

For exposition purposes, it is useful to describe the computations discussed in 5.2.2 in a more generalizable way.

Recall (section 5.2.2) that the entries in the resulting distance matrix were computed as

$$s_{ij} = 1 - \frac{|CS_i \cap CS_j|}{|CS_i \cup CS_j|} ,$$

where CS_i stands for the "core set" of node o_i as defined in 5.1.2. New expressions for $|CS_i \cap CS_j|$ and $|CS_i \cup CS_j|$ can be derived as follows:

- For each $o_i \in O$ (the original set of nodes), define a vector $v_i = (a_{i1}, a_{i2}, \ldots, a_{in})$, i.e., the row of the adjacency matrix corresponding to node o_i; recall that $a_{ii} = 1 \; \forall \; i$, here.

- It is clear now that $|CS_i \cap CS_j| = v_i \cdot v_j$, the inner product of vectors v_i and v_j; recall that their components are either zero or one.

- Furthermore, since $|CS_i \cup CS_j| = |CS_i| + |CS_j| - |CS_i \cap CS_j|$, and

$|CS_i| = v_i \cdot v_i$, it follows that

$|CS_i \cup CS_j| = v_i \cdot v_i + v_j \cdot v_j - v_i \cdot v_j$

Thus, the expression for s_{ij} can now be written

$$s_{ij} = 1 - \frac{v_i \cdot v_j}{v_i \cdot v_j + v_j \cdot v_j - v_i \cdot v_j} \quad .$$

Note that:

1) When $v_i \equiv v_j$, $s_{ij} = 0$;

2) When $v_i \cdot v_j = 0$ (i.e., v_i and v_j are orthogonal), $s_{ij} = 1$.

With this formalization, by analogy, a given similarity matrix S can be transformed into a normalized distance matrix $D(d_{ij})$ as follows:

- Define n vectors s_{i*}, $i = 1, \ldots, n$, as follows:

  $s_{i*} = (s_{i1}, s_{i2}, \ldots, s_{in})$.

- Compute D's entries as:

$$d_{ij} = 1 - \frac{s_{i*} \cdot s_{j*}}{s_{i*} \cdot s_{i*} + s_{j*} \cdot s_{j*} - s_{i*} \cdot s_{j*}} \quad .$$

Unfortunately, the distances d_{ij} computed this way cannot be shown to be metrics, due to the fact that the components of vectors v_{i*} are not binary as they were with v_i. (In general, they do not meet condition (4) of section 3). An adjustment can be made, however, to overcome this problem. If we define the entries in a distance matrix $D^*(d^*_{ij})$ as:

$$d^*_{ij} = 1 - \frac{s_{i*} \circledcirc s_{j*}}{s_{i*} \circledcirc s_{j*} + s_{j*} \circledcirc s_{j*} - s_{i*} \circledcirc s_{j*}}, \text{ where}$$

$$s_{i*} \odot s_{j*} = \sum_{k=1}^{n} [\min(s_{ik}, s_{jk})]^2 ,$$

it can be shown that the entries in D^* are metrics (see Appendix III).

Actually, assuming that $s_{ij} \geq 0 \quad \forall i,j$, as it would be in most similarity matrices, the above definition can be changed to

$$s_{i*} \odot s_{j*} = \sum_{k=1}^{n} \min(s_{ik}, s_{jk}) ,$$

and the entries in D^* would still be metrics.

This adjustment intuitively follows the strategy used in the graph case: Given two vectors v_i and v_j, we compute $v_i \cdot v_j$ as

$$v_i \cdot v_j = \sum_{p=1}^{n} s_{ip} s_{jp} ,$$

but since $s_{ip} \in \{0,1\} \quad \forall i,p$, what we really do is the following: If both nodes o_i and o_j <u>are</u> related to o_p, we set $s_{ip}s_{jp}$ to 1, but to 0 if either one (or both) are not related to o_p. In the new setting, the definition of $s_{i*} \odot s_{j*}$ tries to do the same: since the s_{ij}'s are similarity measures, when we come across a pair of objects (o_i, o_j) not equally similar to a third o_p, we are "conservative" and use only the smaller similarity measure.

The similarity matrix D^* computed as above can be used in any cluster analysis algorithm.

The following observations need to be made at this point:

a) Most similarity matrices constructed for cluster analysis purposes pay no attention to their main diagonal entries, since they are not used in cluster analysis algorithms. The normalization procedure described above <u>requires</u> that these entries be consistent with the remaining ones, because they <u>are</u> used in the normalization process. More concretely, in a given similarity matrix $S(s_{ij})$, it should be true that $s_{ii} = s_{jj} \quad \forall i,j$

and that $s_{ii} \geq \max \{s_{ij} \in S, i \neq j\}$. With this provision, if any d^*_{ij} computed as indicated above turns out to be zero, objects o_i and o_j can be collapsed into a new one, (and thus clustered together), as was done in the graph case. If these conditions are not met by the entries in S, the normalization procedure can produce misleading results: For example, assume that $s_{ij} = s_{ii} = s_{jj} = \min \{s_{ij} \in S\}$, and that $s_{i*} = s_{j*}$; this would lead to $d^*_{ij} = 0$ and thus to one cluster containing o_i and o_j, but o_i and o_j are the most dissimilar pair of objects in O, since $s_{ij} = \min \{s_{ij} \in S\}$!

b) "Collapsable" objects can still result. Since such objects will end up in the same cluster, the normalization procedure also gives more "apparent structure" to the original data.

c) The discussion above assumed that S was a similarity matrix. If it was a dissimilarity one, we can still apply the procedure by first defining $B = \max (s_{ij})$, then computing $S'(s'_{ij})$ by letting $s'_{ij} = B - s_{ij}$, and working with S'. Note that for S' to meet the conditions set forth in (a) above, S should be such that $s_{ii} = s_{jj} \quad \forall$ i,j and $s_{ii} \leq \min \{s_{ij} \in S, i \neq j\}$ (i.e., a given object shouldn't be more dissimilar -- less similar -- to itself than to another object).

d) By letting

$$d^{*'}_{ij} = 1 - d^*_{ij} \ ,$$

the resulting matrix would be a normalized similarity matrix. So, the procedure described above can produce a normalized $\{^{\text{similarity}}_{\text{dissimilarity}}\}$ matrix out of a $\{^{\text{similarity}}_{\text{dissimilarity}}\}$ one.

5.3.2.1.-Iterative computation of distance matrices.

The nature of the strategy proposed above for computing normalized $\{^{\text{distance}}_{\text{similarity}}\}$ matrices from $\{^{\text{distance}}_{\text{similarity}}\}$ matrices is such that nothing precludes us from using it iteratively, i.e.: computing a normalized distance matrix, then using it as input to compute a second one, and so on. This possibility brings up interesting questions. For example:

- Does the iterative application of the procedure eventually converge to a "stable" matrix?

- If collapsable objects can result at each iteration, would this iterative procedure behave as a hierarchical clustering algorithm?

These questions are, at least in principle, more of a theoretical interest than of a practical one, given that such an iterative procedure is likely to be slow (at each iteration an entire new matrix is computed). Nevertheless they are relevant in order to gain insight about how well the distance matrices computed as proposed convey the overall structure of the set of objects under analysis.

Although we haven't approached the above questions from an analytical viewpoint, we have conducted some experiments which seem to indicate that the answer to both questions is yes.

We illustrate this by means of a few examples below.

Prior to introducing these examples, the following observation needs to be made:

· The nature of metric distance matrices is such that it precludes the result $d_{ij} = 0$ with $i \neq j$ from occurring (recall that one metric property precisely states that $d_{ij}=0 \iff i = j$). Therefore, we would never be capable of collapsing objects in the iterative procedure that we are about to explore. However, it turns out that, as the iterations proceed, there are distances (entries in the computed distance matrix) which keep decreasing and taking values more and more close to zero. For our purposes here, we allow the specification of a parameter ε, close to zero, to be used in conjunction with the iterative procedure as follows: whenever a computed distance d_{ij} is less than ε, objects o_i and o_j are clustered together.

Some Examples

Example 1

The first example was chosen to be trivial from a decomposition viewpoint; its main thrust is to show that the iterative procedure behaves as expected in the simplest case imaginable. Consider the graph in Fig. 13:

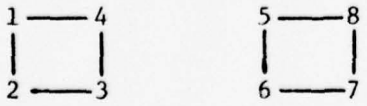


Figure 13

It is not completely connected and consists of two disjoint subgraphs. We would expect a decomposition procedure to result in the partition {1,2,3,4}, {5,6,7,8}. Below we show that the iterative procedure does indeed obtain this

partition; in addition, there are other interesting observations to be made.

The adjacency matrix corresponding to the graph in Fig. 13 is shown below. Note that this matrix can be viewed as its initial similarity matrix; it is interesting to compare it with the current similarity matrix after a number of iterations.

	1	2	3	4	5	6	7	8
1	1	1	0	1	0	0	0	0
2	1	1	1	0	0	0	0	0
3	0	1	1	1	0	0	0	0
4	1	0	1	1	0	0	0	0
5	0	0	0	0	1	1	0	1
6	0	0	0	0	1	1	1	0
7	0	0	0	0	0	1	1	1
8	0	0	0	0	1	0	1	1

After four iterations, the similarity matrix looks as follows:

	1	2	3	4	5	6	7	8
1	1.00	0.94	0.94	0.94	0.00	0.00	0.00	0.00
2	0.94	1.00	0.94	0.94	0.00	0.00	0.00	0.00
3	0.94	0.94	1.00	0.94	0.00	0.00	0.00	0.00
4	0.94	0.94	0.94	1.00	0.00	0.00	0.00	0.00
5	0.00	0.00	0.00	0.00	1.00	0.94	0.94	0.94
6	0.00	0.00	0.00	0.00	0.94	1.00	0.94	0.94
7	0.00	0.00	0.00	0.00	0.94	0.94	1.00	0.94
8	0.00	0.00	0.00	0.00	0.94	0.94	0.94	1.00

After four more iterations, it appears like:

	1	2	3	4	5	6	7	8
1	1.00	1.00	1.00	1.00	0.00	0.00	0.00	0.00
2	1.00	1.00	1.00	1.00	0.00	0.00	0.00	0.00
3	1.00	1.00	1.00	1.00	0.00	0.00	0.00	0.00
4	1.00	1.00	1.00	1.00	0.00	0.00	0.00	0.00
5	0.00	0.00	0.00	0.00	1.00	1.00	1.00	1.00
6	0.00	0.00	0.00	0.00	1.00	1.00	1.00	1.00
7	0.00	0.00	0.00	0.00	1.00	1.00	1.00	1.00
8	0.00	0.00	0.00	0.00	1.00	1.00	1.00	1.00

where entry values have been rounded to the nearest two decimal numbers. If we compare this matrix with the original similarity (adjacency) matrix, it is obvious that the graph structure has been made much more apparent.

Taking $\varepsilon = 0.001$, the expected partition is obtained after three more iterations. Furthermore, the matrix becomes completely stable. After this iteration, the current distance matrix has only two rows and two columns, corresponding to the two subgraphs. It looks like:

	1	2
1	0.00	1.00
2	1.00	0.00

It is interesting to note that the entries other than the main diagonal entries have the maximum possible value (1, recall that distance matrices are normalized), indicating that the two subgroups identified are

completely independent for our purposes; this is intuitively correct given that they were disjoint from the start.

Example 2

The next example takes a less trivial case. We use the graph in Fig. 6, reproduced in Fig. 14.

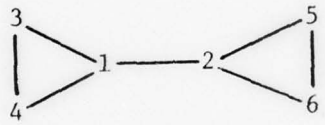


Figure 14

Taking an ε value of 0.001, the distance matrix becomes stable after 20 iterations. Two subgraphs are identified, {1,3,4} and {2,5,6} . The final distance matrix is shown below:

	1	2
1	0.00	0.43
2	0.43	0.00

Note that in contrast with the preceding example, entries not in the main diagonal take values below 1., indicating that the two subgroups were not originally disjoint.

Example 3

We saw in the previous example that when the identified subgraphs are not originally disjoint, the final distance matrix reflects this circumstance by setting non-main diagonal entries to values less than 1. A natural question to ask is whether this intuitive behavior continues when the coupling between identified subgraphs varies in importance. This example and the next explore this question.

The graph in Fig. 15 differs from that in Fig. 14 in the number of links between the two eventual subgraphs.

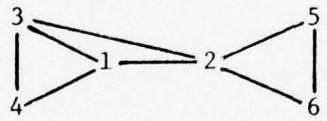


Figure 15

With an ϵ value of 0.001 as before, the iterative procedure produces the same partition {1,3,4}, {2,5,6} after 16 iterations, but the final distance matrix is as follows:

	1	2
1	0.00	0.13
2	0.13	0.00

Note that compared with that in Example 2, it behaves as expected.

Example 4

The graph in Fig. 16 represents another step towards more coupling between subgraphs.

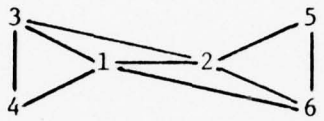


Figure 16

After 18 iterations, the same partition is identified but the final distance matrix indicates that the two subgraphs are closer together:

	1	2
1	0.00	0.08
2	0.08	0.00

* * *

The preceding examples have shown that the proposed iterative procedure works in an intuitively appealing way. They raise some questions, however. For example:

(a) What about ε values?

(b) Does the procedure always end up with <u>two</u> subgraphs (subsets)?

We have explored these questions working with several additional examples. Our results are summarized below:

(a) The chosen ε value doesn't seem to matter much as long as it is teasonably close to zero. For instance, all the examples above reach the same final result with ε = 0.01, 0.001 and 0.0001. Of course, the smaller ε, the larger the number of iterations needed to achieve stability in the distance matrix. Moreover, as ε increases, the intermediate clustering may vary although the final result remains the same. If ε is increased substantially, the final result may change significantly. For example, taking ε = 0.09 in example 4 above (see the final distance matrix in which the largest entry is 0.08) would result in clustering the entire graph together.

(b) The procedure doesn't always end up with two subgraphs, although this is the most common result. Exceptions have been experienced when:

1. There are more than two disjoint subgraphs initially; as many subgraphs are identified whatever the <u>ε</u> value employed.

2. The graph is so compact that it is not partitioned at all. This may depend upon ε, as suggested above. However, our experience indicates that if no partition is identified with a reasonably small ε, the initial graph or set is indeed too compact to be partitioned in any way.

The fact that more than two subgraphs (subsets) are not identified unless disjoint subgraphs exist at the beginning is somewhat disturbing. We have observed the following behavior, however: Whenever the result is in the form of two subgraphs, either one of them or both are such that taken independently are not partitioned by the procedure. This suggests that the following strategy can be used to partition a given graph using this iterative approach:

1) Set "current graph" to be the initial graph;
2) Use the iterative technique to partition it;
3) If the outcome is no partition, save the "current graph" as a final subgraph. Go to 5;
4) Otherwise, put the two (or more) resulting subgraphs in a "subgraph pool".
5) Check the "subgraph pool". If it is empty, stop: The subgraphs saved at step (3) represent a partition. If not, take one subgraph from the pool. Make it the "current graph". Go to 2.

We have used this strategy with a number of graphs; the results have been encouraging.

In summary, two main comments can be made about this approach:

- It requires a single parameter: ε .
- It seems to be very robust with respect to different "reasonable" ε values.
- The final result gives a "feel" for how interrelated the identified subsets are, through the final distance matrix.

5.3.3. - The strategy of section 5.3.1 revised.

Now that we have developed a generalization of the normalization procedure originally introduced for the binary matrix case, we are in a position where the strategy of section 5.3.1 for obtaining initial partitions in a non-agglomerative cluster analysis algorithm can be revised in a very appealing way. The revision permits us to eliminate the need for the threshold parameter T, whose choice was not obvious; also, by using a percentage parameter p of the type suggested at the end of section 5.1.2, we eliminate the need for k.

If we keep in mind that one possible generalization of the expression $s_{i*} \cdot s_{j*}$ (in the terminology of the preceding section) is

$$s_{i*} \circledcirc s_{j*} = \sum_{k=1}^{n} \min(s_{ik}, s_{jk}) ,$$

the following procedure can be used (note the parallelism with that in section 5.1.2), where we assume $S(s_{ij})$ to be a similarity matrix (see Appendix IV for a way of converting dissimilarity matrices into similarity ones so that the procedure below is generally applicable):

Given p, a percentage parameter ,

0) $J = 0$.

1) Compute $c_i = \sum_{\substack{q=1 \\ q \neq i}}^{n} s_{iq}$, $i=1, \ldots, n$, and

$$|CS_i| = c_i + s_{ii} , \quad i=1, \ldots, n.$$

2) Compute $c_0 = \max_{i=1,\ldots,n} (c_i)$;

Select all objects o_j such that

$$c_j \geq \frac{p}{100} c_0 .$$

Assume there are k such nodes; without loss of generality, let them be the nodes $o_1, \ldots, o_k$.

If $k = n$, set $J = J+1$, $KESU(J) = O$ and stop.

3) For all $o_i \in \{o_1, \ldots, o_k\}$ compute

$$|KS_i| = \sum_{q=1}^{n} \min(s_{iq}, \max_{\substack{j=1 \\ j\neq 1}}^{k} (s_{jq})).$$

4) Select o_r such that

$$|KS_r| = \min_{j=1}^{k} (|KS_j|).$$

5) $J = J + 1$,

If $|KS_r| = |CS_r|$, set $KESU(J) = O$ and stop;

else, set $KESU(J) = o_r \cup \pi$, where

$$\pi = \{o_t | o_t \text{ s.t. } s_{rt} > \max_{\substack{j=1 \\ j\neq r}}^{k} (s_{jt}),\ t=1, \ldots, n\} .$$

6) Recompute O: $O = O - KESU(J)$; if $|O| = 0$, stop.

7) Recompute S:

$$S: \left\{ s_{ij}\ s_{ij} = \left\{ \begin{array}{l} \text{old } s_{ij} \text{ iff } o_i, o_j \in O \\ \text{mark it "nonexistent" otherwise} \end{array} \right\} \right\}$$

Go to 1.

Two examples of this procedure are illustrated below. They refer to the same problem but use different matrices. The first step in both is

to transform a distance matrix into a similarity matrix as indicated in Appendix IV. Although this is not really necessary, since we can always change "max" to "min", ">" to "<" and vice versa in the above procedure for it to be applicable to dissimilarity matrices, we do it for clarity. A parameter p = 80 is assumed in both.

A) The matrix in Fig. 8, corresponding to the "minimum path" metric distances of the graph in Fig. 6 is transformed to the similarity matrix shown in Fig. 17 using the strategy of Appendix IV.

i \ j	1	2	3	4	5	6
1	3	2	2	2	1	1
2	2	3	1	1	2	2
3	2	1	3	2	0	0
4	2	1	2	3	0	0
5	1	2	0	0	3	2
6	1	2	0	0	2	3

Fig. 17

The procedure then proceeds as follows:

0) $J = 0$

1) $c_1 = c_2 = 8, \quad c_3 = c_4 = c_5 = c_6 = 5.$

$|cs_1| = |cs_2| = 11, \quad |cs_3| = |cs_4| = |cs_5| = |cs_6| = 8.$

2) $c_0 = \max(8,5) = 9$;

since $0.8*8 = 6.4 > 5$, $k=2$, o_1 and o_2 are selected.

3) $|KS_1| = |KS_2| = 8 \quad (< 11)$

4) $o_r = o_1$ (arbitrarily)

5) $J = 1$,

$|KS_1| < |CS_1|$,

$KESU(1) = \{o_1, o_3, o_4\}$.

6) $O = \{o_2, o_5, o_6\}$.

7) Delete rows and columns correponding to o_1, o_3, o_4.

1) $c_2 = c_5 = c_6 = 4$; $\quad |CS_2| = |CS_5| = |CS_6| = 7$.

2) $c_0 = 4$,

o_2, o_5, o_6 are selected.

Since $k = n$ here, $J = 2$ and

$KESU(2) = \{o_2, o_5, o_6\}$,

and the procedure terminates.

Thus, the graph in Fig. 6 is partitioned in the best possible way.

B) For the same graph in Fig. 6, we can use the noramlized matrix in Fig. 10, which was derived in a different way. Transforming it into a similarity matrix as before yields the matrix in Fig. 18.

i \ j	1	2	3(4)	5(6)
1	1	0.34	0.75	0.17
2	0.34	1	0.17	0.75
3(4)	0.75	0.17	1	0
5(6)	0.17	0.75	0	1

Fig. 18

Operating upon this matrix, the procedure unfolds as follows:

0) $J = 1$.

1) $c_1 = c_2 = 1.26$, $c_{3(4)} = c_{5(6)} = 0.92$.

$|CS_1| = |CS_2| = 2.26$, $|CS_{3(4)}| = |CS_{5(6)}| = 1.92$.

2) $c_0 = \max(1.26, 0.92) = 1.26$

since $0.8*1.26 = 1.008 > 0.92$, o_1 and o_2 are selected.

3) $|KS_1| = |KS_2| = 1.02$ (< 2.26)

4) $o_r = o_1$ (arbitrarily)

5) $J = 1$,

$|KS_1| < |CS_1|$,

$KESU(1) = \{o_1, o_{3(4)}\}$

6) $O = \{o_2, o_{5(6)}\}$

7) Delete rows and columns corresponding to o_1 and $o_{3(4)}$.

1) $c_2 = c_{5(6)} = 0.75$; $|CS_2| = |CS_{5(6)}| = 1.75$.

2) $c_0 = 0.75$, and o_2 and $o_{5(6)}$ are selected.
Since k = n here, J = 2 and
KESU(2) = $\{o_2, o_{5(6)}\}$,
and the procedure terminates.

Note that the obtained result, taking into account the preliminary clustering performed as a byproduct while computing the normalized dissimilarity matrix of Fig. 10, coincides with the result in (A) above, so that the best partition for the graph in Fig. 6 is again identified.

6. - Other approaches and problems.

Section 5 assumed that the graph under decomposition was "undirected" (i.e., its adjacency matrix was symmetric). When the links of a graph bear a certain direction, the graph is called "directed," or sometimes "digraph." Note that in the context of our problem a directed graph would mean that a certain requirement may be related to another, but that the opposite is not necessarily true. It turns out that the problem of digraph decomposition is easier to solve than the general graph decomposition problem. Several effective heuristic approaches have been proposed for this purpose (see, for instance, [Haralick 74] or [Boesch & Gimpel 77]).

A formal treatment of the digraph decomposition problem can be found in [Kevorian & Snoek 71]. They propose a methodology that decomposes digraphs in a hierarchical fashion; the resulting hierarchy explicitly points out how the identified subgraphs interact, as a consequence of the "precedence" characteristic implicit in a digraph. In addition, a very nice correspondence is shown: the digraph decomposition problem is equivalent to the decomposition of a set of objects characterized by a number of attributes in the following way:

(a) The number of objects is the same as the number of attributes, and

(b) Objects' respresentations in terms of attributes take the form of binary vectors (i.e., in a sense, an attribute is either "relevant" or not to a given object).

Limitation (a) makes this approach unsuited to our problem, but (b) suggests a way of assessing interdependencies among objects (system requirements in our case) that may be worth investigating.

7. - Summary and implications.

The discussion above has described some cluster analysis and graph decomposition techniques, and has pointed out strategies that make possible the application of the former to graph decomposition and vice versa. In this section we summarize these techniques and strategies and comment on their application.

The discussion in sections 4 and 5 can be summarized as follows:

1) It is possible to put a graph decomposition problem in the framework of cluster analysis. Since many cluster analysis techniques exist, this can be useful to solve graph decomposition problems.

2) Certain heuristics that are useful in graph decomposition have a clear counterpart in cluster analysis. In particular, they can be employed to identify initial partitions in non-agglomerative cluster analysis methods.

3) Normalized similarity (or dissimilarity) matrices can be derived for both graph decomposition and cluster analysis problems. In addition, the resulting matrices display the structure of the decomposition problem better than the original data (e.g., "collapsable" objects are made apparent).

4) Agglomerative cluster analysis algorithms are usually fast but make early commitments that are never revised and which can lead to undesirable results.

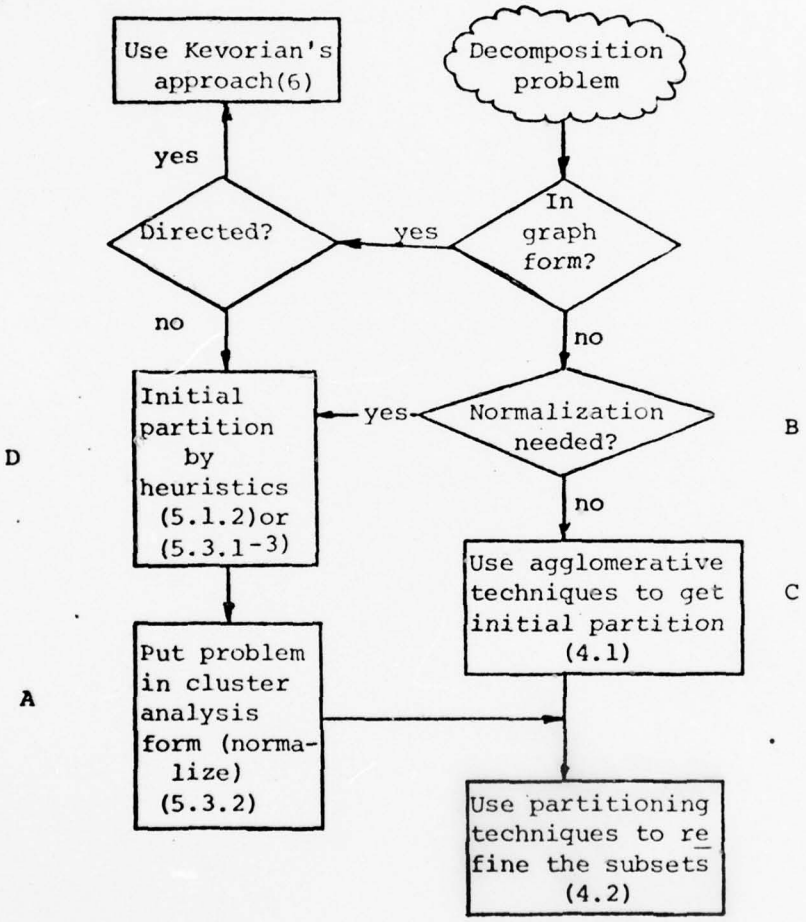

Use Kevorian's approach(6)
Decomposition problem
yes
Directed?
yes
In graph form?
no
no
Initial partition by heuristics (5.1.2)or (5.3.1-3)
yes
Normalization needed?
B
D
no
Use agglomerative techniques to get initial partition (4.1)
C
Put problem in cluster analysis form (normalize) (5.3.2)
A
Use partitioning techniques to refine the subsets (4.2)

Fig. 19

If a normalization route is taken we advocate the use of the heuristic approach described in sections 5.3.1 and 5.3.3 to identify an initial partition, the reason being that some computations needed for the normalizati procedure can also be used for partitioning purposes.

The output of Fig. 19 will be a decomposition of the original set of objects. In [Andreu & Madnick 77] we proposed to interpret the result intuitively. A more formal approach, which is possible if the initial problem originated from a characterization of each object by means of a set of attributes, is to analyze which of these attributes are predominant in each of the subsets, using factor analysis.

With regard to a methodology for the assessment of interdependencie among pairs of objects (requirements), it seems that working with a meaningfu set of attributes gives wide choices with respect to applicable decomposition niques. In addition, proceeding this way would reduce the number of assessmer be made (na as opposed to n^2, where n is the number of objects and a the number of attributes; we are assuming $n > a$). Significant improvement over a simple yes/no assessment would be achieved even with the simplest attribute-oriented approach (like assessing only the relevance or non-relevance of each attribute for each object).

APPENDIX I

For the discussion in this Appendix, we assume that the graph (O,A) is connected (i.e., there are no disjoint subgraphs). This assumption is reasonable because if there were disjoint subgraphs they could be treated separately, i.e., they would be completely independent of one another for our purposes.

With this provision, we show that if we define

s_{ij} = Number of links in the minimum path from o_i to o_j,

then it is true that:

a) $s_{ij} = 0 \Leftrightarrow i = j \quad \forall i,j,$

b) $s_{ij} \geq 0 \quad \forall i,j,$

c) $s_{ij} = s_{ji} \quad \forall\ i,j,$ and

d) $s_{ij} \leq s_{ik} + s_{kj} \quad \forall i,j,k.$

We proceed as follows:

a) It is obvious that $s_{ii} = 0 \quad \forall i,$ so that

$i = j \Rightarrow s_{ij} = 0 \quad \forall i,j.$

Furthermore, $s_{ij} = 0$ implies that we can travel from node o_i to node o_j without traversing any link. This is only possible if i = j. So,

$s_{ij} = 0 \Rightarrow i = j \quad \forall i,j,$

which completes the proof.

b) $s_{ij} \geq 0$ follows immediately from the definition of s_{ij} (i.e., "number of links" can not be negative).

c) Since the adjacency matrix A is symmetric, so is S:

$s_{ij} = s_{ji} \quad \forall i,j.$

d) To show that

$$s_{ij} \leq s_{ik} + s_{kj} \quad \forall\, i,j,k,$$

we consider two cases:

i) o_k is in some minimum path from o_i to o_j.

If this is the case, since both s_{ik} and s_{kj} are minimum path lengths, there is no better way of going from o_i to o_j than going from o_i to o_k in s_{ik} steps, then from o_k to o_j in s_{kj} steps, i.e.:

$$s_{ij} = s_{ik} + s_{kj}.$$

ii) o_k is not in a minimum path from o_i to o_j.

This means that $s_{ik} + s_{kj}$ is the length of a path from o_i to o_j which is not minimum, i.e.:

$$s_{ij} < s_{ik} + s_{kj}.$$

APPENDIX II

Given three nodes o_i, o_j and $o_k \in O$, with associated core sets CS_i, CS_j and CS_k, we defined s_{ij}, s_{ik} and s_{kj} as follows:

$$s_{ij} = 1 - \frac{|CS_i \cap CS_j|}{|CS_i \cup CS_j|} \quad , \qquad s_{ik} = 1 - \frac{|CS_i \cap CS_k|}{|CS_i \cup CS_k|} \quad , \text{ and}$$

$$s_{kj} = 1 - \frac{|CS_k \cap CS_j|}{|CS_k \cup CS_j|} \quad .$$

Here, we show that these quantities meet conditions 2, 3 and 4 of section 3, i.e., that

a) $s_{ij} \geq 0 \ \forall i,j$,

b) $s_{ij} = s_{ji} \ \forall i,j$, and

c) $s_{ij} \leq s_{ik} + s_{kj} \ \forall i,j,k$.

We proceed as follows:

a) Since for any pair of sets CS_i and CS_j it is always true that

$$|CS_i \cup CS_j| \geq |CS_i \cap CS_j|, \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (1)$$

it is obvious that

$$\frac{|CS_i \cap CS_j|}{|CS_i \cup CS_j|} \leq 1, \ \forall i,j.$$

Thus, $s_{ij} \geq 0 \ \forall i,j$.

Furthermore, since $|CS_i \cap CS_j| \geq 0$[*] for any i,j, (1) also implies that

$$s_{ij} \leq 1 \ \forall i,j.$$

---

[*] Throughout the discussion in this Appendix, it is useful to keep in mind that $|X| \geq 0$, whatever the set X is.

b) Since both set union and intersection are commutative operations, it follows that

$$s_{ij} = s_{ji} \quad \forall i,j.$$

c) To show that

$$s_{ij} \leq s_{ik} + s_{kj} \quad \forall i,j,k, \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (2)$$

we will consider three cases that cover all the possible values for s_{ij} ($0 \leq s_{ij} \leq 1$, see (a) above).

i) $s_{ij} = 0$.

Since $s_{ik}, s_{kj} \geq 0$ (see (a) above), (2) follows immediately in this case.

ii) $s_{ij} = 1$.

By definition of s_{ij}, this implies that

$|CS_i \cap CS_j| = 0$, i.e., $CS_i \cap CS_j = \phi$, so that

$$|CS_i \cap CS_k| + |CS_k \cap CS_j| \leq |CS_k| \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (3)$$

(because since CS_i and CS_j are disjoint, elements in CS_k can only belong to <u>either</u> CS_i <u>or</u> CS_j).

Then, $s_{ik} + s_{kj} < 1$ ($= s_{ij}$) is impossible, because it implies

$$1 - \frac{|CS_i \cap CS_k|}{|CS_i \cup CS_k|} + 1 - \frac{|CS_k \cap CS_j|}{|CS_k \cup CS_j|} < 1, \quad \text{or}$$

$$\frac{|CS_i \cap CS_k|}{|CS_i \cup CS_k|} + \frac{|CS_k \cap CS_j|}{|CS_k \cup CS_j|} > 1 \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (4)$$

If we now realize that

$$\left.\begin{aligned} |CS_i \cup CS_k| &\geq |CS_k| \quad \text{and} \\ |CS_k \cup CS_j| &\geq |CS_k| \quad , \end{aligned}\right\} \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (5)$$

it follows that if (4) holds, so does

$$\frac{|cs_i \cap cs_k|}{|cs_k|} + \frac{|cs_k \cap cs_j|}{|cs_k|} > 1, \qquad \text{i.e.,}$$

$$\frac{|cs_i \cap cs_k| + |cs_k \cap cs_j|}{|cs_k|} > 1,$$

which contradicts (3).

iii) $s_{ij} = 1 - a, \quad 0 < a < 1.$

This implies, by definition of s_{ij}, that

$$|cs_i \cap cs_j| = a|cs_i \cup cs_j| \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (6)$$

Moreover, $s_{ij} > s_{ik} + s_{kj}$ in this case implies that

$$\frac{|cs_i \cap cs_k|}{|cs_i \cup cs_k|} + \frac{|cs_k \cap cs_j|}{|cs_k \cup cs_j|} > 1+a \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (7)$$

We consider two subcases:

α) $|cs_k| \geq |cs_i \cup cs_j| \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (8)$

Since $|cs_i \cap cs_k| \leq |cs_i|$, and $|cs_k \cap cs_j| \leq |cs_j|$, it is true that

$$|cs_i \cap cs_k| + |cs_k \cap cs_j| \leq |cs_i| + |cs_j| = |cs_i \cup cs_j| + |cs_i \cap cs_j|, \qquad \ldots\ldots (8a)$$

i.e., by (6),

$$|cs_i \cap cs_k| + |cs_k \cap cs_j| \leq (1+a)|cs_i \cup cs_j| \qquad \ldots\ldots\ldots\ldots (9)$$

Furthermore, by (5), (7) implies

$$\frac{|cs_i \cap cs_k| + |cs_k \cap cs_j|}{|cs_k|} > 1+a, \qquad \text{i.e.,}$$

$$|cs_i \cap cs_k| + |cs_k \cap cs_j| > (1+a)|cs_k|, \qquad \text{or, by (8),}$$

$$|cs_i \cap cs_k| + |cs_k \cap cs_j| > (1+a)|cs_i \cup cs_j|,$$

which contradicts (9).

β) $|CS_k| < |CS_i \cup CS_j|$.

This subcase requires a more careful proof. The following arguments simplify it:

- $s_{ij} > s_{ik} + s_{kj}$ cannot hold unless CS_k has elements in coomon with <u>both</u> CS_i and CS_j. To see this, recall that by definition of s_{ik}, $CS_i \cap CS_k = \phi \Longrightarrow s_{ik} = 1$, so that since $s_{ij} \leq 1$ the above inequality is impossible regardless of the value of s_{kj}. Therefore, we need only to consider the case in which $CS_i \cap CS_k \neq \phi$ <u>and</u> $CS_k \cap CS_j \neq \phi$.

- If $s_{ij} > s_{ik} + s_{kj}$ does not hold when $CS_k \cup (CS_i \cup CS_j) = CS_i \cup CS_j$ (i.e, when CS_k has no elements outside $CS_i \cup CS_j$), it will not hold when CS_k has elements outside $CS_i \cup CS_j$. To see this, let

CS_k^1 represent the part of CS_k inside $CS_i \cup CS_j$, and

CS_k^2 represent the part of CS_k outside $CS_i \cup CS_j$, i.e.:

$$CS_k^1 \cup CS_k^2 = CS_k,$$

$$CS_k^1 \cap CS_k^2 = \phi,$$

$$CS_k^1 \cup (CS_i \cup CS_j) = CS_i \cup CS_j, \quad \text{and}$$

$$CS_k^2 \cap (CS_i \cup CS_j) = \phi.$$

Then, it is apparent that

$$|CS_i \cup CS_k| = |CS_i \cup CS_k^1| + |CS_k^2|,$$

$$|CS_k \cup CS_j| = |CS_k^1 \cup CS_j| + |CS_k^2|,$$

$$|CS_i \cap CS_k| = |CS_i \cap CS_k^1|, \quad \text{and}$$

$$|CS_k \cap CS_j| = |CS_k^1 \cap CS_j|,$$

so that (7) can be written as follows:

$$\frac{|CS_i \cap CS_k^1|}{|CS_i \cup CS_k^1| + |CS_k^2|} + \frac{|CS_k^1 \cap CS_j|}{|CS_k^1 \cup CS_j| + |CS_k^2|} > 1+a,$$

from where we see that if does not hold for $|CS_k^2| = 0$, it certainly won't hold for $|CS_k^2| > 0$.

Consequently, in what follows we limit our attention to proof that (7) is impossible when

$$CS_k \cap CS_i \neq \phi, \qquad CS_k \cap CS_j \neq \phi \qquad \text{and} \qquad CS_k \cup (CS_i \cup CS_j) = CS_i \cup CS_j.$$

To do this, we rewrite (7) as follows:

$$\frac{|CS_i \cap CS_k|}{|CS_i \cup CS_k|} + \frac{|CS_k \cap CS_j|}{|CS_k \cup CS_j|} > \frac{|CS_i| + |CS_j|}{|CS_i \cup CS_j|}, \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots (10)$$

and consider the following cases:

I) $|CS_k| = |CS_i \cup CS_j| - n, \qquad n > 0,$

where the n elements belong to $CS_i \cap CS_j$ (recall that since we assume $a > 0$, $CS_i \cap CS_j$ is non-empty).

The situation is depicted in Fig. AII-1.

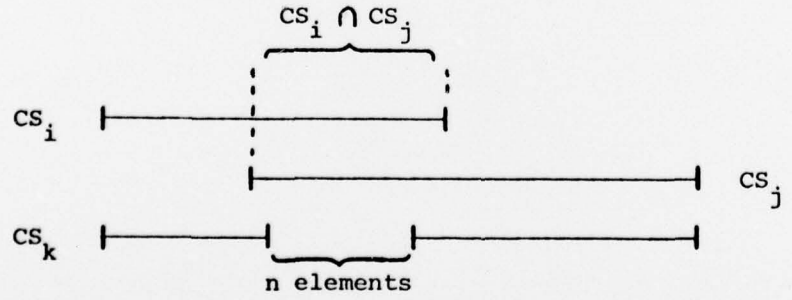


Fig. AII-1

In this case, (10) would read

$$\frac{|CS_i| - n}{|CS_i \cup CS_j| - n} + \frac{|CS_j| - n}{|CS_i \cup CS_j| - n} > \frac{|CS_i| + |CS_j|}{|CS_i \cup CS_j|}, \quad \text{i.e.:}$$

$$|CS_i||CS_i \cup CS_j| - n|CS_i \cup CS_j| + |CS_j||CS_i \cup CS_j| - n|CS_i \cup CS_j| >$$

$$|CS_i||CS_i \cup CS_j| - n|CS_i| + |CS_j||CS_i \cup CS_j| - n|CS_j|,$$

i.e.:

$$n\{|CS_i| + |CS_j| - 2|CS_i \cup CS_j|\} > 0,$$

or, since $|CS_i \cup CS_j| = |CS_i| + |CS_j| - |CS_i \cap CS_j|$,

$$n\{2|CS_i \cap CS_j| - |CS_i| - |CS_j|\} > 0,$$

which is impossible because $|CS_i| \geq |CS_i \cap CS_j| \leq |CS_j|$ and $n > 0$.

II) $|CS_k| = |CS_i \cup CS_j| - n_i - n_j, \qquad n_i, n_j > 0,$

where the n_i elements belong to CS_i but not to $CS_i \cap CS_j$, and the n_j belong to CS_j but not to $CS_i \cap CS_j$.

The situation is depicted in Fig. AII-2.

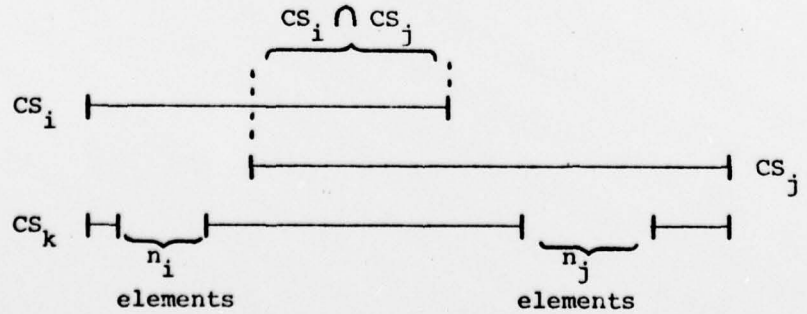


Fig. AII-2

In this case, (10) would read

$$\frac{|CS_i| - n_i}{|CS_i \cup CS_j| - n_j} + \frac{|CS_j| - n_j}{|CS_i \cup CS_j| - n_i} > \frac{|CS_i| + |CS_j|}{|CS_i \cup CS_j|} \qquad \ldots\ldots\ldots\ldots\ldots (11)$$

(note that this expression reduces to the previous case when $n_i = n_j$).

Expanding,

$$\{|CS_i| - n_i\}\{|CS_i \cup CS_j| - n_i\}|CS_i \cup CS_j| + \{|CS_j| - n_j\}\{|CS_i \cup CS_j| - n_j\}|CS_i \cup CS_j| >$$

$$\{|CS_i| + |CS_j|\}\{|CS_i \cup CS_j| - n_j\}\{|CS_i \cup CS_j| - n_i\}.$$

For convenience, let

$$|CS_i| = I, \quad |CS_j| = J, \quad |CS_i \cup CS_j| = A \qquad \ldots\ldots\ldots\ldots\ldots (12)$$

We then have:

$$IA^2 - n_iIA - n_iA^2 + n_i^2A + JA^2 - n_jJA - n_jA^2 + n_j^2A >$$

$$IA^2 - n_iIA - n_jIA + n_in_jI + JA^2 - n_iJA - n_jJA + n_in_jJ, \qquad \text{i.e.:}$$

$$n_iA(J + n_i - A) + n_jA(I + n_j - A) - n_in_j(I + J) > 0 \qquad \ldots\ldots\ldots\ldots\ldots (13)$$

By (12), it is now apparent that (see Fig. AII-2 and recall that we assumed that neither n_i nor n_j belong to $CS_i \cap CS_j$)

$$J + n_i - A = |CS_j| + n_i - |CS_i \cup CS_j| \leq 0,$$

$$I + n_j - A = |CS_i| + n_j - |CS_i \cup CS_j| \leq 0, \qquad \text{and obviously}$$

$$-n_in_j(|CS_i| + |CS_j|) < 0,$$

so that (13), and hence (11) and (7), are impossible.

III) $|CS_k| = |CS_i \cup CS_j| - n_i - n_j - n, \qquad n_i, n_j, n > 0,$

where n_i and n_j are the same as before and the n elements belong to $CS_i \cap CS_j$.

The situation is depicted in Fig. AII-3.

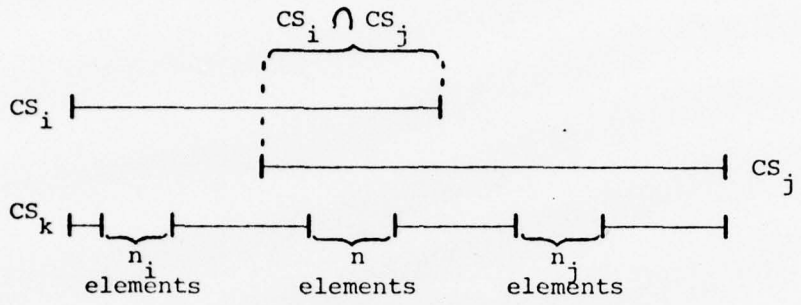


Fig. AII-3

In this case, (10) would read

$$\frac{|CS_i| - n_i - n}{|CS_i \cup CS_j| - n_j} + \frac{|CS_j| - n_j - n}{|CS_i \cup CS_j| - n_i} > \frac{|CS_i| + |CS_j|}{|CS_i \cup CS_j|} \quad .$$

Comparing this equation with (11), since $n > 0$, it is obvious that if (11) does not hold neither will this one. This completes the proof for subcase β.

APPENDIX III

Here we show that the distances d^*_{ij} defined as

$$d^*_{ij} = 1 - \frac{s_{i*} \circledcirc s_{j*}}{s_{i*} \circledcirc s_{i*} + s_{j*} \circledcirc s_{j*} - s_{i*} \circledcirc s_{j*}} \quad , \qquad \ldots\ldots\ldots\ldots\ldots\ldots (1)$$

where

$$s_{i*} \circledcirc s_{j*} = \sum_{k=1}^{n} [\min(s_{ik}, s_{jk})]^2 \quad , \qquad \ldots\ldots\ldots\ldots\ldots\ldots (2)$$

and where the s_{ik}'s are entries in a similarity matrix with the properties discussed in section 5.3.2, meet the matric conditions of section 3.

We proceed as follows:

a) $d^*_{ij} = 0$ implies that (by (1) and (2))

$$\sum_{k=1}^{n} [\min(s_{ik}, s_{jk})]^2 = \sum_{k=1}^{n} s_{ik}^2 + \sum_{k=1}^{n} s_{jk}^2 - \sum_{k=1}^{n} [\min(s_{ik}, s_{jk})]^2 \quad , \text{ i.e.,}$$

$$\sum_{k=1}^{n} [\min(s_{ik}, s_{jk})]^2 = \frac{\sum_{k=1}^{n} s_{ik}^2 + \sum_{k=1}^{n} s_{jk}^2}{2} \quad ,$$

which in turn implies

$$s_{ik} = s_{jk} \quad , \; k = 1, \ldots, n.,$$

for otherwise

$$\sum_{k=1}^{n} \min(s_{ik}, s_{jk})^2 < \frac{\sum_{k=1}^{n} s_{ik}^2 + \sum_{k=1}^{n} s_{jk}^2}{2} \qquad \ldots\ldots\ldots\ldots\ldots\ldots (3)$$

Thus,

$$d^*_{ij} = 0 \Rightarrow s_{i*} = s_{j*} \; .$$

As in the graph case, it may happen that

$d^*_{ij} = 0$ with $i \neq j$,

but we take care of this circumstance by collapsing objects o_i and o_j, so that for all practical purposes,

$$d^*_{ij} = 0 \Rightarrow i = j \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (4)$$

Furthermore, if $i = j$ obviously $d^*_{ij} = 0$. This, together with (4) yields

$$d^*_{ij} = 0 \Leftrightarrow i = j \text{ (collapsing nodes if necessary)} \ldots\ldots\ldots\ldots\ldots\ldots (5)$$

b) Equation (3) readily implies that

$d^*_{ij} > 0$ if $s_{i*} \neq s_{j*}$,i.e., by (5),

$d^*_{ij} \geq 0 \quad i,j.$

Furthermore, it is apparent that $s_{i*} \circledcirc s_{j*} \geq 0$, so that

$d^*_{ij} \leq 1 \quad i,j.$

c) By definition of $\circledcirc$ (see (2) above), and since "min" is a commutative operation, it follows that

$$d^*_{ij} = d^*_{ji} \quad \forall i,j.$$

d) To show that

$$d^*_{ij} \leq d^*_{ik} + d^*_{kj} \quad \forall i,j,k, \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (5a)$$

we can follow exactly the same strategy as in point c) of Appendix II. We will not repeat it all here; we will only show that the same inequalities employed there hold true here. (In what follows, we refer to equation i of Appendix II by II-i):

- Equation II-3, in the new notation, would read

$$s_{i*} \odot s_{k*} + s_{k*} \odot s_{j*} \leq s_{k*} \odot s_{k*} \quad , \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (6)$$

i.e., by (2),

$$\sum_{p=1}^{n} [\min(s_{ip}, s_{kp})]^2 + \sum_{p=1}^{n} [\min(s_{kp}, s_{jp})]^2 \leq \sum_{p=1}^{n} s_{kp}^2$$

Since obviously

$$\min^2(s_{ip}, s_{kp}) \leq s_{kp}^2 \quad \forall\ i,k,p \text{ , because we assume } s_{kp} \geq 0, \quad \ldots\ldots (7)$$

(6) holds true.

- Equation(s) II-5 would read

$$\sum_{p=1}^{n} [\min(s_{ip}, s_{kp})]^2 \geq \sum_{p=1}^{n} s_{kp}^2 \quad \forall\ i,k \ ,$$

which by (7) is also true.

- Equation II-8a would read

$$\underbrace{\sum_{p=1}^{n} [\min(s_{ip}, s_{kp})]^2}_{A} + \underbrace{\sum_{p=1}^{n} [\min(s_{kp}, s_{jp})]^2}_{B} \leq \underbrace{\sum_{p=1}^{n} s_{ip}^2}_{C} + \underbrace{\sum_{p=1}^{n} s_{jp}^2}_{D} \ ,$$

which also by (7) holds here, since $A \leq C$ and $B \leq D$.

- The proof corresponding to subcase β in Appendix II, finally, can be followed step by step in the generalized setting.

APPENDIX IV

Here we show a very simple way of converting a dissimilarity matrix $D(d_{ij})$ into a similarity one $S(s_{ij})$. By simply letting

$$s_{ij} = A - d_{ij},$$

where $A = \max (d_{ij} \in D)$,

the entries in S will behave as similarity coefficients. In particular, if the entries in D are metrics (see section 3), then the entries in S will have the following properties:

1) $s_{ij} = A$ iff $i = j$, because $d_{ij} = 0$ iff $i = j$.

2) $s_{ij} = s_{ji} \quad \forall\ i,j$, because $d_{ij} = d_{ji} \quad \forall\ i,j$.

3) $s_{ij} \geq 0 \ \forall\ i,j$

4) $s_{ij} \leq A \ \forall\ i,j$

> because of the definition of A.

REFERENCES

[Anderberg 73]:
Anderberg, M. R.: "Cluster Analysis for Applications," Academic Press, New York, 1973.

[Andreu & Madnick 77]:
Andreu, R. C. and S. E. Madnick: "A systematic approach to the design of complex systems: application to DBMS design and evaluation," CISR Report No.32, Sloan School of Management, M.I.T., March 1977.

[Astrahan 70]:
Astrahan, M. M.: "Speech analysis by clustering, or the hyperphoneme method," Stanford Articifial Intelligence Project Memo AIM-124, AD 709067, Stanford University, 1970.

[Boesch & Gimpel 77]:
Boesch, F. T. and J. F. Gimpel: "Covering the points of a digraph with point-disjoint paths and its application to code optimization," Journal of the Association for Computing Machinery, 24-2, April 1977.

[Choffray 77]:
Choffray, J. F.: Unpublished doctoral thesis, M.I.T. Sloan School of Management, April 1977.

[Curry 76]:
Curry, D. J.: "Some statistical considerations in clustering with binary data," Multivariate Behavioral Research, April 1976.

[Deo 74]:
Deo, N.: "Graph Theory with Application to Engineering and Computer Science," Prentice Hall, New York, 1974.

[Freeman 76]:
Freeman, P.: "The context of design," in "Tutorial on Software Design Techniques," P. Freeman, ed., IEEE Catalogue No.76CH1145-2C.

[Haralick 74]:
Haralick, Robert M.: "The diclique representation and decomposition of binary relations," Journal of the Association for Computing Machinery, 21-3, July 1974.

[Hartigan 75]:
Hartigan, J.: "Clustering Algorithms," Wiley Interscience, 1975.

[Hubert 74]:
Hubert, L. J.: "Some applications of graph theory to clustering," Psychometrika 39 (September 1974).

REFERENCES [cont'd]

[Kevorian & Snoek 71]:
Kevorian, A. K. and J. Snoek: "Decomposition in large-scale systems: theory and applications of structural analysis in partitioning, disjoining and constructing hierarchical systems," in "Decomposition of Large-Scale Systems," D. M. Himmelblau, ed., North-Holland Publishing Co., 1973.